Ching-Hua Chuan* and Elaine Chew†

*University of North Florida
School of Computing
1 UNF Drive Mathews Building
Jacksonville, Florida 32224-2645 USA
c.chuan@unf.edu
†Queen Mary, University of London
School of Electronic Engineering
and Computer Science
Mile End Road
London E1 4NS UK
elaine.chew@eecs.qmul.ac.uk

Generating and Evaluating Musical Harmonizations That Emulate Style

In this article we present a hybrid approach to the design of an automatic, style-specific accompaniment system that combines statistical learning with a music-theoretic framework, and we propose quantitative methods for evaluating the results of machine-generated accompaniment. The system is capable of learning accompaniment style from sparse input information, and of capturing style over a variety of musical genres. Generating accompaniments involves several aspects, including choosing chords, determining the bass line, arranging chords for voicing, instrumentation, etc. In this article we focus on harmonization: selecting chords for melodies, with an emphasis on style. Given exemplar songs as MIDI melodies with corresponding chords labeled as text, the system uses decision trees to learn the melody–chord relations shared among the songs. Markov chains on the neo-Riemannian framework are applied to model the likelihood of chord patterns. Harmonization is generated in a divide-and-conquer manner: Melody fragments that strongly imply certain triads are designated as checkpoints that are in turn connected by chord progressions generated using the Markov model. Chord subsequences are then refined and combined to form the final sequence.

We propose two types of measures to quantify the degree to which a machine-generated accompaniment achieves its style emulation goal: one based on chord distance, and the other based on the statistical metrics entropy and perplexity. Using these measures, we conduct two sets of experiments using Western popular songs. Two albums by each of three artists (Green Day, Keane, and Radiohead), for a total of six albums, are used to evaluate the proposed system. The first set of experiments are inter-system tests that compare the output of the proposed system with that of the Temperley-Sleator (T-S) Harmonic Analyzer, a rule-based system, and that of a naïve harmonization system. The results show that the hybrid system produces harmonizations that have more chords identical to those in the original song, and that are more "consistent" with the original. Consistent harmonizations have more chord phrases—a sequence of chords that harmonizes a (vocal) melody phrase—that are frequently observed in the original, as indicated by the lower entropy and perplexity values. The second set of experiments consists of intra-system comparisons that test the effect of training data (one versus many) on style emulation effectiveness, and tests that investigate the performance of each system component separately. The results show that the system with the single-song training set generates, on average, more chords that are identical to the original than systems trained on all songs in the album. The results also demonstrate the robustness of the music-theoretic framework against erroneous information. Although the system is evaluated using popular music, it can be generalized to produce style-specific harmonization for other music genres where Western music theory is applied.

Computer Music Journal, 35:4, pp. 64–82, Winter 2011
© 2011 Massachusetts Institute of Technology.

Introduction

Automatic generation of music in a specific style has been a focus for computational music research since the early days of computing. Researchers have designed systems that imitate musical styles ranging from baroque to jazz. The results can be impressive, according to subjective professional

opinion, but the applications remain limited. Most systems are optimized for a particular style which, once established, cannot be easily adapted to other styles. Users of these existing systems play passive roles as listeners, and have no way of defining which elements of the style they seek in the compositions that are generated. One of our goals is to create a flexible style-emulating system that can imitate any style, while allowing users to specify their preferences.

In this article, we describe our automatic style-specific accompaniment system, named ASSA, which focuses on the harmonization of a given melody. Given pitch and chord data as input, this system is capable of automatically harmonizing user-created melodies, with proper harmonic resolutions, in a style that the system learns from only a few examples. Based on the given examples representing the style the user desires, the system identifies the features important to that style and generates chords to any new melody in a manner such that the resulting song sounds stylistically similar to the examples. With the assistance of ASSA, users can compose songs in their preferred style without invoking formal musical terminology, and can affect the style of the output by simply providing the system with examples.

Statistical learning is commonly used for discovering regularities and extracting patterns from examples; such methods have recently been employed in accompaniment generating systems (Ponsford, Wiggins, and Mellish 1998; Lee and Jang 2004). Statistical learning methods require large training sets; when only limited numbers of training examples are available, problems may emerge. Although, on an epic scale, style can refer to the traits common to a complete genre of music, style can also refer to individual idiosyncrasies in the crafting of a single composition. Sparsity of examples is thus unavoidable in style-emulating accompaniment systems: Individual peculiarities, important to specifying a song's style, can be diluted by large numbers of training examples.

As stated previously, we use a hybrid approach in the design of ASSA. The system combines music-theoretic knowledge and statistical learning to model the harmonization choices made by musicians in their representative pieces, then predicts their harmonization decisions for new melodies. This hybrid approach gives the system the flexibility of mimicking various styles from only a few examples, while ensuring that the output follows general music-theoretic rules.

To counter the sparsity-of-examples problem, we use neo-Riemannian operations (NROs) on the tone network (tonnetz) to represent chord transitions, as will be explained later. Neo-Riemannian operators and their combined transforms on the tonnetz allow us to characterize and retrieve common chord patterns in different example contexts by focusing on local relations between chords. This music-theoretic framework ameliorates the effect of extreme examples in sparse data. Using NROs on the tonnetz, we build decision trees to statistically learn melody–chord patterns, and Markov models to represent chord transition likelihoods in the accompaniment.

ASSA generates harmonizations in a divide-and-conquer fashion. The system divides the input melody into segments delineated by bars in which melody notes strongly imply triads. The system then independently generates a sequence of chords for each segment. Chord candidates are chosen using decision trees based on chord-tone assignments learned from the examples. The chord-tone assignments decide which notes of the melody are likely to be chord members, as opposed to non-chord tones (e.g., passing tones). The system concatenates the chord candidates to produce chord sequences for each segment, bearing in mind the NROs between adjacent chords. These sequences are combined, with refinements, to produce the chord progression for the entire melody. This divide-and-conquer approach is designed for finding the most stylistically consistent chord sequence, not for producing harmonizations in real time.

A major hurdle encountered in the design and implementation of style-emulating systems is the evaluation of such systems, or the measurement of the degree to which the output achieves the goal of emulating a particular style. When the emulated style possesses a well-studied form, such as baroque chorale-type harmonizations, music-theoretic rules describing the style can be

used as valid evaluation criteria. In contrast, the standards for evaluating harmonization style in less strict forms, such as pop/rock music, tend to be ambiguous. The complexity derives from the fact that "in rock, . . . melody is allowed to interact with harmony more freely" (Stephenson 2002, p. 75). As a result, evaluation often takes the form of subjective opinion, which may lead to imprecise conclusions that can change dramatically depending on the participants. The second objective of this article, therefore, is to propose general quantitative methods for evaluating the results of machine-generated style-specific harmonization.

Two types of measures are described: The first consists of musical metrics that examine the style as captured by the musical relations between corresponding chords in the original and in the machine-generated harmonizations; the second calculates values of entropy and perplexity to compute the degree of similarity between the music segments generated and the given examples. Based on the proposed measurements, we design two sets of experiments for evaluation. The first, which we label the "inter-system comparisons," compares the style specificity of the system outputs of (1) ASSA, (2) a rule-based harmonization system, and (3) a naïve harmonizer. The second set of experiments, labeled "intra-system comparisons," explores the impact of training-set selection strategies on style emulation and the added benefit of each individual system component. Finally, we describe some case studies in which we examine the training song in detail to determine the factors that will result in the best-case output from the proposed system. As described previously, we conduct the experiments on six well-known pop/rock albums by Green Day, Keane, and Radiohead, and provide detailed statistics on the resulting harmonizations.

The article is organized as follows: We first describe related work on automated accompaniment systems (both with and without style requirements) and their evaluation. The details of ASSA and the proposed evaluation methods are presented next. We then present the experiment design and results for inter- and intra-system comparisons, followed by case studies. Conclusions and topics for discussion are presented at the end of the article.

Related Work

As used here, automatic harmonization is the problem of providing chord sequences that can be elaborated to support a lead melody. Various computing techniques have been used to harmonize a melody in the style of a particular period or composer in Western classical music. Natural language processing (NLP) techniques, such as *n*-gram statistical learning, have been applied to the learning of musical grammars for harmonizing music in a 17th-century style (Ponsford, Wiggins, and Mellish 1998). Evolutionary techniques, such as genetic algorithms, have been proposed and implemented for generating four-part chorales (Phon-Amnuaisuk, Tuwson, and Wiggins 1999; Phon-Amnuaisuk and Wiggins 1999). Hidden Markov models have been utilized in the harmonization of chorales (Allan and Williams 2004) and 16th-century Palestrina-style compositions (Farbood and Schoner 2001). The T-S Harmonic Analyzer (Temperley and Sleator 1999a; Temperley 2002) uses a set of preference rules to harmonize melodies in the canonical Western classical style (Temperley and Sleator 1999b). In these harmonization tasks, each model is created to emulate a particular style that is well documented by a host of music-theoretic rules. These rules can serve as valid guidelines for generating and evaluating the automatically generated harmonization. Although these systems' target styles are well defined, they cannot be changed.

In the popular music realm, the i-Ring system (Lee and Jang 2004) generates accompaniment to any eight-measure melody, based on state transition probabilities calculated from a training set of 150 songs. In its evaluation, ten participants are asked to rate the accompaniment as good, medium, or bad. MySong (Morris, Simon, and Basu 2008) uses hidden Markov models trained on 298 popular songs from genres including pop, rock, rhythm and blues, jazz, and country music. Users can choose between two style-related modes: "jazz," which uses less common triads, or "happy," which selects more major-mode chords. A total of 26 sample accompaniments by MySong were subjectively evaluated by 30 volunteer musicians. Althought the output of these systems fits the melody, the style

captured by the output is comparatively indistinct. These systems do not address the emulation of specific accompaniment styles as embodied in a distinctive song or a particular band's output. Evaluations of the output of these systems have taken only the form of subjective opinion.

The first version of ASSA (Chuan and Chew 2007) introduced the basic system structure and its hybrid features: the music-theoretic framework and statistical learning modules. This version used only 17 attributes to characterize melody–chord patterns. The final chord sequence is generated using root-to-leaf path-finding in a tree structure where branches are grown and pruned before applying conditional probabilities to model chord transitions. The system was subjectively evaluated in two case studies, each of which considered a Radiohead melody with machine-generated accompaniment, trained on three other Radiohead songs. In later research (Chuan and Chew 2008) we introduced systematic and quantitative evaluations of this first ASSA system, comparing the generated harmonizations with the original, which served as the ground truth.

This article presents some of this previous work, and expands on it in a number of ways. Here, the melody–chord patterns are modeled using 73 attributes so as to describe the melodic features more precisely. We apply Lidstone's Law (Lidstone 1920) to smooth the probability distribution generated from the examples. Instead of the tree structure of the first ASSA system, we use Markov models with smoothed probability distributions to generate the chord sequences. For evaluation, we extend the existing measures by adopting two NLP statistical metrics, entropy and perplexity, to examine the generated accompaniments at the segmental level. To study the added benefit of each ASSA component, we test the system with parts of it disabled. We also conduct case studies to investigate the manner in which a training song affects the generated output, and to discover the factors of a training song that can bring about the best-case result from the ASSA system.

Automatic Style-Specific Accompaniment

This section provides an overview and the details of the ASSA system. Figure 1 shows the system and data flow. ASSA takes a user-created melody as input and automatically generates a sequence of chords to harmonize the melody while ensuring that the harmonization choice is stylistically similar to the user-supplied song examples.

Given an input melody, the system first maps pitches into normalized numeric pitch classes, then encodes each melody note using 73 attributes that represent the roles that that note plays in relation to others in the melody. The core of the system includes a module for determining chord tones and another for generating chord progressions, as presented in the dashed boxes in Figure 1.

The chord-tone determination module uses decision trees to train the system on the relation between melody notes and their harmonizing chords in the given examples, and applies the trees thus constructed to the classifying of users' melody notes into chord and non-chord tones. Based on the reported chord tones, melody fragments that strongly imply triads are designated as checkpoints for generating chord progressions.

As stated previously, ASSA takes a divide-and-conquer approach to the generation of chord progressions. The system generates candidate chord sequences between two checkpoints and then combines and refines these sequences to produce the final chord progression for the entire melody. In the chord progression generation module, the neo-Riemannian framework is invoked to represent the transitions between adjacent chords. The final chord progression is created by considering the conditional probabilities of all possible sequences, as represented in the Markov model.

The following sections describe in detail each major ASSA component.

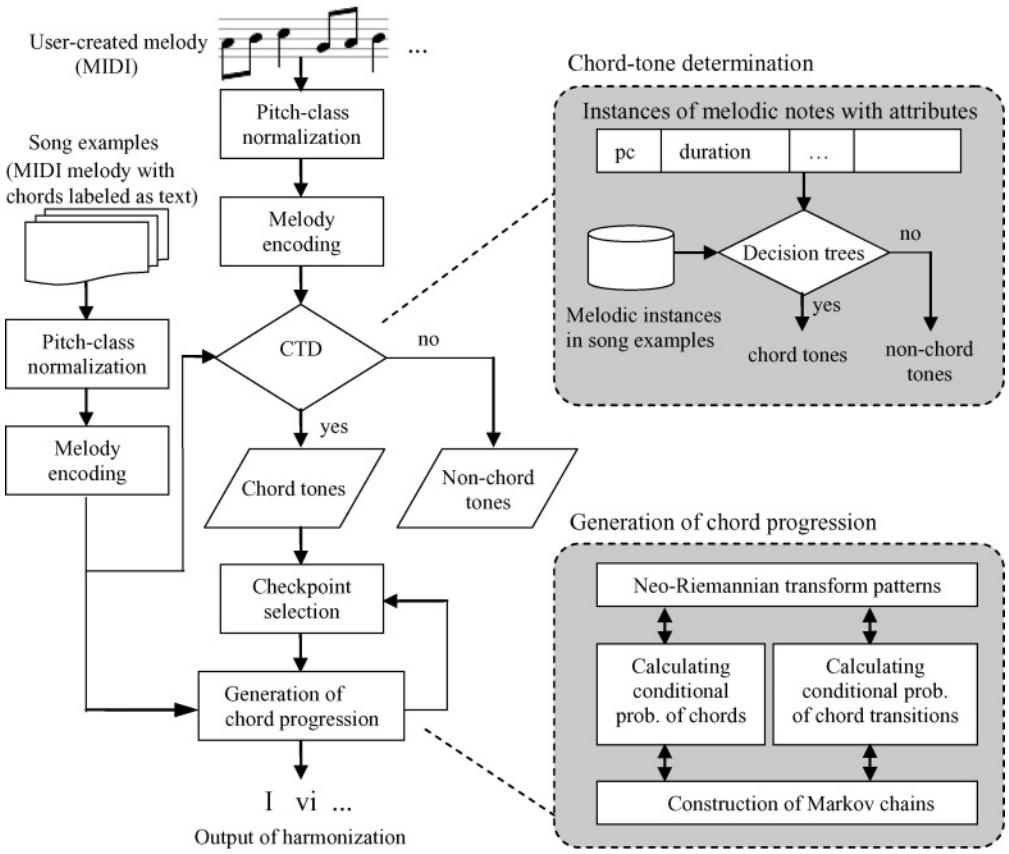
Pre-Processing

There are two steps in the pre-processing of input melodies.

Pitch Class Normalization

The pitches of the input melody, provided by users to be harmonized or to be used as examples, are mapped to their respective pitch class numbers,

Figure 1. The data flow and system components in ASSA.



normalized so that the tonic is zero. For example, given a melody in the key of D, the pitch E_4 maps to the pitch-class number 2. In the MIDI format, the normalized pitch class, PC_i, for a pitch can be calculated by the formula:

$$PC_i = (P_i - P_{tonic}) \text{mod} 12,$$

where P_i represents the MIDI pitch number of the ith pitch (e.g., 62 for pitch D_4) and P_{tonic} represents the MIDI pitch number of the tonic.

Melody Note Encoding

The 73 attributes used for describing each melody note are shown in Table 1. The attributes represent properties of a melody note and its context. These attributes cover a range of hierarchical levels, such as the note's relation to adjacent notes, its position in a bar, its position within a phrase, and its location with respect to the structure (e.g., chorus or verse) of the piece.

The first 50 attributes consider the notes in a melody fragment, where a melody fragment is defined as a series of melody notes that are harmonized by the same chord. These attributes consist of the normalized pitch class and the length of the melody note, and these same attributes of other notes in the same melody fragment. Attributes 51 to 56 represent properties of a note in relation to the ones immediately preceding and following it; for example, whether the adjacent notes are less than two semitones higher or lower. Attribute 57 represents the total number of pitch classes in a melody fragment. Attributes 58 through 65 relate to metric information. Beat strength shows the metric

Table 1. The 73 Attributes for Describing Each Melody Note

No.	*Attribute*	*Description*
1	Pitch class (pc)	Numeric pc, tonic normalized to zero
2	Duration	Note duration (in beats) in melody fragment
3–14	Note *i* semitones above	Does the melody fragment contain a note *i*(= 0. . . 11) semitones above this note?
15–26	Duration(note *i* semitones above)	Duration of note *i*(= 0. . . 11) semitones above
27–38	Note *i* semitones below	Does the melody fragment contain a note *i*(= 0. . . 11) semitones below this note?
39–50	Duration(note *i* semitones below)	Duration of note *i*(= 0. . . 11) semitones below
51	Interval (*left*) ≤ 2	Note approached by 2 or fewer half steps?
52	$2 <$Interval (*left*) ≤ 5	Preceding interval within (2,5] half steps?
53	Interval (*left*) ≥ 5	Preceding interval larger than 5 half steps?
54	Interval (*right*) ≤ 2	Note left by 2 or fewer half steps?
55	$2 <$Interval (*right*) ≤ 5	Successive interval within (2,5] half steps?
56	Interval (*right*) ≥ 5	Successive interval larger than 5 half steps?
57	Number of pcs	Total number of pcs in melody fragment
58–61	Metrical strength level *l*	Does this note have metrical strength level *l*(= 1. . . 4)?
62–65	Beat *j*	Note on beat *j*(= 1. . . 4)?
66	Fragment number in phrase	Fragment index in phrase
67–71	Fragment position S within phrase	Does fragment occur at position S (= {*start*, *middle*, *end*, *final*})?
72	Phrase number in song	Phrase index in song
73	Verse/chorus/others	Is note in verse, chorus or others?

strength (Volk 2002; Chew, Volk, and Lee 2005) of the pulse on which the melody note resides, and the beat attribute records the metric beat according to the time signature. The remaining attributes provide information about note position with respect to the bar, phrase, and section (e.g., verse, chorus).

In this study, published lead sheets are used for system evaluation. A lead sheet contains information on melody, chords, bars, key signature, time signature, and lyrics. In our study, notes in the melody are first encoded as MIDI. To generate the 73 attributes for describing the context of a melody note, other information in the lead sheet is used. For instance, the key signature is used to convert notes into normalized pitch classes. Most of the other attributes are generated automatically based on the information given in the lead sheet. Some attributes, on the other hand, require manual annotation. For example, the labeling of phrase position depends partly on the lyrics, but mostly on an individual's perception. Verse or chorus labeling is another example where manual annotation is required if no such information is given in the lead sheet. For evaluation purposes, as long as the labeling scheme is consistent, the way in which a label is selected will not affect the system's performance.

Chord-Tone Determination

The chord-tone determination module is designed to learn the relationship between melody and harmony in the given song examples, and to ascertain that relationship in new user-defined or test melodies in the style of the given examples.

The relationship between melody notes and chord harmonies can be expressed in terms of binary classification: If the note is a member of the chord, then it is classified as a chord tone; otherwise, it is labeled as a non-chord tone. The varieties of non-chord tones utilized in compositions are part of the unique musical vocabularies of composers, and have been used to analyze and identify individual

composers' styles (Huang and Chew 2005). Classical music theory textbooks such as Kostka and Payne (2003) prescribe rules for determining chord tones and non-chord tones. In ASSA, we apply machine learning to the given examples to build the chord-tone classification model. We believe that the rules thus constructed can capture accompaniment style in various genres, including and beyond classical music.

We use a decision tree to determine chord tones. Decision trees can represent complex if-then-else sequences, handle discrete and continuous data, and reduce the effect of noise in the training set by calculating information gain. They are easily readable to the human eye and map readily to music-theoretic rules. The decision tree module is responsible for selecting chord tones from each melody fragment in order to learn, based on the examples provided, a particular band or songwriter's typical harmonization decisions. In the learning stage, the module represents each melody note by its 73 attributes, as described in Table 1, then calculates the effect of the attributes on chord-tone classification. In the testing stage, the module classifies a note as a chord tone or non-chord tone based on its 73 attributes. The guitar chords in the lead sheet provide the ground truth.

We use the C4.5 decision tree algorithm with subtree raising pruning (Quinlan 1993; Mitchell 1997; Witten and Frank 2005), and use tenfold cross validation to solve the problem of the lack of training data. The parameters for pruning and cross validation are experimentally optimized on the eleven songs in the British band Keane's album *Hopes and Fears*. The output of the chord-tone determination module is a list of notes that should be considered chord tones in each test piece melody fragment. Given the list of chord tones, a few possible chords can then be chosen as harmonization candidates.

Checkpoint Selection

Given a list of chord tones for each melody fragment, we designate fragments in which chord tones strongly imply triads as checkpoints. A checkpoint fragment may be a bar consisting of chord tones with the root and third, or third and fifth, of a candidate chord present. We can assign chords to the checkpoint fragments with strong evidence of a particular harmony choice independently of the fragments with less evidence of harmonicity.

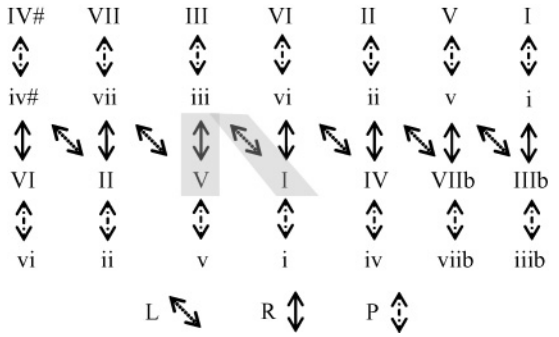
The setting up of checkpoints divides the chord progression generation task into smaller sections of chord series settings. Instead of determining the chord sequence for the entire melody all at once, we generate a suitable series of chords between each pair of adjacent checkpoints. For fragments with weak harmonic evidence, we use the probability distribution over the neo-Riemannian framework to assign chords based on adjacent chord selections. Each chord series is modeled as a Markov chain, with the likelihood of the series in question calculated from conditional probabilities learned from the example songs. Finally, all chord series are combined, and the most plausible combination selected to give the chord progression for the entire melody. When joining two adjacent chord series where the common checkpoint can be harmonized with more than one valid candidate chord, we make the combined series a single Markov chain, and choose the chord that results in the longest and highest probability path, eliminating all others with probability below a predefined threshold.

Neo-Riemannian Operations

Music theorists have used NROs to characterize triad progressions, particularly those of the late Romantic period (Lewin 1987; Cohn 1996, 1997; Childs 1998; Hook 2002). More recently, NROs have been used to analyze harmonic patterns and voice-leading in pop/rock music (Kochavi 2002; Capuzzo 2004). In ASSA, NROs form the basis for our representation of chord transitions.

Figure 2 (after Capuzzo 2004) shows the neo-Riemannian chord space, where chords are represented by upper case (major) and lower case (minor) Roman numerals. Chords are connected by three types of NROs: the leading-tone exchange (L), relative (R), and parallel (P) operations. Each operation preserves the identity of two tones common to the adjacent chords, and ensures parsimonious

Figure 2. Neo-Riemannian operations in chord space.



voice leading when transitioning from one chord to the next. Transitions between two chords can be described by single or compound NROs; for example, the transition from the V triad to the I triad can be represented as the compound operation RL (the shaded path in Figure 2).

The neo-Riemannian framework provides a solution to the sparse data problem in the training of the system. NROs allow the system to extend the number of possible chord patterns based on very limited numbers of examples while ensuring that extended patterns contain voice-leading patterns consistent with the original. For example, the V to I pattern can be extended to other transitions, such as I to IV and vi to iii, as long as the transition follows an RL NRO sequence. In this way we assign probabilities to all RL sequences, instead of calculating the conditional probabilities for the V to I, I to IV, and vi to iii patterns separately.

Markov Chains

Using the neo-Riemannian framework, we can represent all possible chord series between two checkpoints with a Markov model. We assume that the choice of a chord only depends on the chord before it and the melody fragment that it harmonizes. By learning from the examples the probabilities of all NROs between adjacent chords, we can compute the probability of each chord series using the chain rule. Suppose we have a series of n chords, $\{C_1, \ldots, C_n\}$, where each chord is indexed by its fragment number. The probability that this chord series occurs can be expressed as:

$$\begin{aligned} P(C_1, \ldots, C_n) &= P(C_1)P(C_2|C_1)\ldots P(C_n|C_{n-1}) \\ &= P(C_1)P(NRO_{1,2})\ldots P(NRO_{n-1,n}), \end{aligned}$$

where $NRO_{i-1,i}$ is the neo-Riemannian operation between chord C_{i-1} and C_i.

Structurally, music is often analyzed at the phrase level, which consists of series of chords. We are not only concerned with the choice of individual chords or the relation between adjacent chords, but also in capturing higher-level musical effects such as tension and relaxation. To model phrase-level effects, we calculate the probability of the chord series, taking into consideration the phrase information. To this end, we modify the previous equation to include the phrase position information of each fragment:

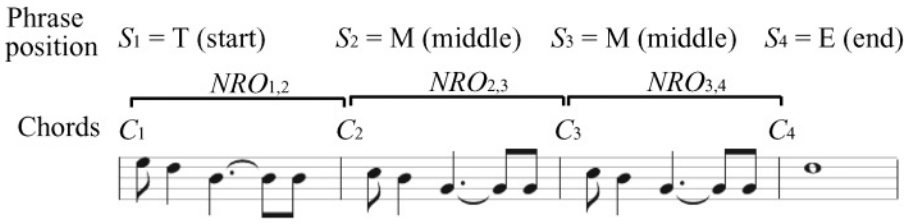
$$\begin{aligned} P(C_1, \ldots, C_n|S_1, \ldots, S_n) &= P(C_1|S_1)P(C_2|C_1, S_1, S_2)\ldots \\ &\quad P(C_n|C_{n-1}, S_{n-1}, S_n) \\ &= P(C_1|S_1)P(NRO_{1,2}|S_1, S_2)\ldots \\ &\quad P(NRO_{n-1,n}|S_{n-1}, S_n), \end{aligned}$$

where S_i is the phrase position of fragment i, which falls into one of four possible categories: start (T), middle (M), ending (E), and final (F), as listed in Table 1, and n is the total number chords in the series. (A fragment is marked with an ending (E) if it is the end of a line in the lyrics, and with a final (F) if it is the end of a stanza or chorus, typically characterized by a final cadence.) For example, the probability that the compound NRO, LR, occurs from a phrase start fragment at index a to a middle phrase fragment at index a+1, $P(NRO_{a,a+1} = LR \mid S_a = T, S_{a+1} = M)$, can be calculated from the examples as follows:

$$\begin{aligned} &P(NRO_{a,a+1} = LR|S_a = T, S_{a+1} = M) \\ &\qquad = \frac{P(NRO_{a,a+1} = LR, S_a = T, S_{a+1} = M)}{P(S_a = T, S_{a+1} = M)}. \end{aligned}$$

To overcome the problem of over-fitting caused by sparse data, we apply Lidstone's Law (Lidstone 1920) to smooth the probability distribution generated from the training examples. Let $N(C_i, S_i)$ be the number of instances where chord C_i occurs at

Figure 3. An example illustrating the relation between chords, phrase positions, and neo-Riemannian operations in a Markov chain.



phrase position S_i in the examples. The conditional probability $P(C_i|S_i)$ is then smoothed using Lidstone's Law as follows:

$$P(C_i|S_i) = \frac{N(C_i, S_i) + \lambda}{N(S_i) + N_c \times \lambda},$$

where N_C is the total number of chords (or melody fragments) in the examples, λ is set to 0.5 as suggested in Manning and Schütze (1999), and $i = 1, \ldots, N_C$. Similarly, the conditional probability of neo-Riemannian operations $NRO_{i-1,i}$ from phrase position S_{i-1} to S_i can be smoothed as follows:

$$P(NRO_{i-1,i}|S_{i-1}, S_i) = \frac{N(NRO_{i-1,i}, S_{i-1}, S_i) + \lambda}{N(S_{i-1}, S_i) + N_C \times \lambda},$$

where $i = 2, \ldots, NC$.

Consider the musical phrase in Figure 3. The chord sequence consists of four chords, $C_1, \ldots, C_4$, in phrase positions $S_1, \ldots, S_4$, respectively. Neo - Riemannian operations are used to describe the relations between adjacent chords; for example, $NRO_{1,2}$ represents the transition from C_1 to C_2. The conditional probability of the chord sequence given the phrase positions is calculated as follows:

$$\begin{aligned} & P(C_1, C_2, C_3, C_4, |S_1, S_2, S_3, S_4,) \\ = \; & P(C_1|S_1)P(C_2|C_1, S_1, S_2)P(C_3, |C_2, S_2, S_3) \\ & P(C_4|C_3, S_3, S_4) \\ \cong \; & P(C_1|S_1)P(NRO_{1,2}|S_1, S_2)P(NRO_{2,3}|S_2, S_3) \\ & P(NRO_{3,4}|S_3, S_4), \end{aligned}$$

and each term in the equation is then smoothed using Lidstone's Law.

Evaluation Metrics for Style-Specific Harmonization

The proposed methods for evaluating the results of machine-generated style-specific harmonization are presented in this section. The evaluation of automated harmonization systems, and the degree to which they emulate a style, has been based primarily on subjective opinion. To quantify style similarity between machine-generated and original harmonizations, we propose two types of measures: one based on musical relations between corresponding chord pairs, and another that examines harmonizations at the phrase level using NLP statistical measurements. Table 2 lists the metrics for assessing machine-generated harmonization. The next two sections present the details for computing these metrics.

Summary Statistics for Chord-Pair Comparisons

The first two metrics in Table 2 are measurements representing musical relations between chords. The first metric, labeled "same chord," records the percentage of chords generated that are identical to the original. The second metric, labeled "chords in grid," presents the percentage of generated chords that are closely related to the original on the chord map (described in the next paragraph). When comparing a machine-generated harmonization with the original, corresponding chords that are in the same position (melody fragment) of the two harmonizations are selected as a pair, and the relation between these two chords in the pair is examined. The

Table 2. Evaluation Metrics

Name	*Meaning*
same chords	percentage of generated chords identical to original
chords-in-grid	percentage of generated chords in chord map
average entropy	average entropy of chord phrases in harmonization
average perplexity	average perplexity of chord phrases in harmonization

percentages are calculated after comparing all the corresponding chord pairs in the two harmonizations.

The relation between two closely related chords can be described using a *chord map*. A chord map, shown in Figure 4, is a nine-cell grid in which each cell represents a closely related chord with respect to the center chord. The Identity (I) is the center chord; the closely related chords adjacent to I are: the Dominant (D), Subdominant (S), Parallel (P), and Relative (R) chords. Four chords, resulting from combination operations DR, SR, DP, and SP, form the other cells in the grid. If a machine-generated chord is present anywhere on a chord map that is centered on the chord from the original accompaniment (the ground truth), it will be considered an acceptable result.

Additional quantitative metrics for evaluating style-emulating harmonizations can be found in Chuan and Chew (2008).

Entropy and Perplexity of Chord Phrases

Entropy and perplexity are two measurements commonly used in NLP and speech recognition to evaluate the quality of constructed speech and language models. In information theory, entropy is a measure of uncertainty—the greater the entropy, the more uncertain or unpredictable the random variable. According to Manning and Schütze, "if a model captures more of the structure of a language, then the entropy of the model should be lower" (Manning and Schütze 1999, p. 73).

Formally, the entropy of a random variable X is defined as:

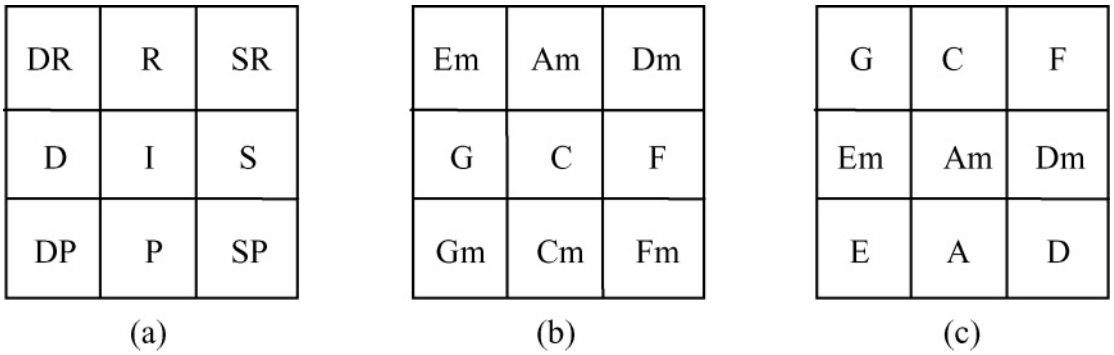
$$H(X) = \sum_{x \in X} P(X = x) \log \frac{1}{P(X = x)}$$
$$= -\sum_{x \in X} P(X = x) \log P(X = x),$$

where $P(X = x)$ is the probability of an instance, x, of the random variable X. For our purpose we consider X a chord in the harmonization.

As mentioned in the construction of Markov chains, we are not only concerned with the choice of individual chords, but also with the relation between adjacent chords, i.e., the chord transitions. We are interested in examining harmonizations at the level of chord phrases, which form the equivalent of sentence units in language. A harmonization can be described by the types of chord phrases within it; the probability distribution of the chord phrases represents the composer's preference. The preferred chord "idioms" can be then considered a contributing factor to the accompaniment style. Therefore, we can examine the similarity between a machine-generated harmonization and the original by looking at the likelihood that the generated chord phrases would appear in the harmonization, based on the chord phrase idioms in the original style.

Given a machine-generated harmonization consisting of chord phrases, we calculate the cross entropy and perplexity of each generated phrase based on the probability distribution of chord phrases in the original harmonization. These values represent how unexpected each generated chord phrase is according to the chord idioms in the target style. We then calculate the average of cross entropy

Figure 4. Chord map: (a) the nine close chords; (b) chords in grid for C major triad; and (c) chords in grid for A minor triad.



and perplexity among all chord phrases, and use the average as the similarity measure for the whole generated harmonization with respect to the original.

Suppose a phrase, R, consists of the n-chord sequence $C_1 C_2 \ldots C_n$. Based on the asymptotic equipartition property (Manning and Schütze 1999), we can use the following expression to calculate cross entropy, which serves as an approximation to the entropy of phrase R:

$$H(R) \cong -\frac{1}{n} \log P(R = C_1 C_2 \ldots C_n = c_1, c_2 \ldots c_n),$$

where C_i is the random variable of a chord and c_j represents an instance. The approximation requires that the modeled process be a stationary ergodic one. We assume that the harmonization style does not change over the time under consideration (i.e., that the process is stationary) and that each chord is allowed to transition to any other chord (i.e., that the process is ergodic).

Recall that $S_k \in \{ T, M, E, F \}$ is the phrase position of fragment k, whose chord is C_k. Assuming that the choice of a chord only depends on the previous one, we can apply the chain rule to the previous equation to obtain:

$$\begin{aligned} H(R) &\cong -\frac{1}{n} \log P(C_1 C_2 \ldots C_n = c_1 c_2 \ldots c_n) \\ &= -\frac{1}{n} \log \left[P(C_1 = c_1 | S_1 = T) \prod_{i=2}^{n} P(C_i = c_i | \right. \\ &\qquad \left. S_i = s_i, C_{i-1} = c_{i-1}, S_{i-1} = s_{i-1}) \right] \\ &= -\frac{1}{n} \left[\log P(C_1 = c_1 | S_1 = T) + \sum_{i=2}^{n} \log P(C_i = c_i | \right. \\ &\qquad \left. S_i = s_i, C_{i-1} = c_{i-1}, S_{i-1} = s_{i-1}) \right]. \end{aligned}$$

For a harmonization A consisting of m phrases $R_1 R_2 \ldots R_m$, we can calculate the average cross entropy of the phrase sequence as:

$$H(A) = \frac{1}{m} \sum_{i=1}^{m} H(R_i).$$

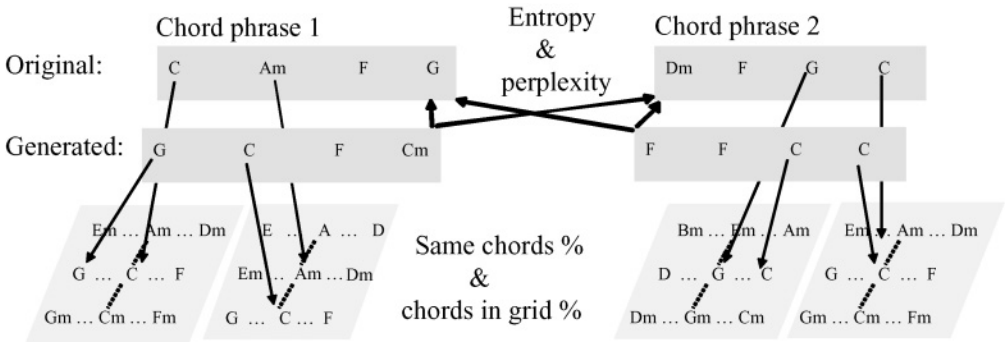
The *perplexity* of the harmonization can be calculated by applying the formula:

$$Perp(A) = \frac{1}{m} \sum_{i=1}^{m} 2^{H(R_i)}.$$

A perplexity value of p means that at each melody fragment, there are on average p possible chord-sequence choices.

Figure 5 illustrates the process of calculating these metrics. In the figure, major chords are denoted using upper-case letters, and minor chords are labeled with an upper-case letter and a lower-case "m." To calculate the same-chords and chords-in-grid percentages, we align the two harmonizations and conduct vertical chord-pair comparisons using the chord map. To calculate the average entropy and perplexity between the two harmonizations, we first take each chord phrase from the generated harmonization and compare it horizontally against all chord phrases in the original to calculate the cross entropy and perplexity for that phrase. We then

Figure 5. Illustration of the evaluation of harmonizations using vertical and horizontal comparisons. Thick arrows indicate comparisons (entropy and perplexity) between chord phrases; pairs of thin arrows indicate comparisons (same-chords and chords-in-grid) between an original chord and the corresponding generated chord.



compute the average cross entropy and perplexity across all phrases. By using both types of evaluation metrics (i.e., summary statistics of chord pairs as well as entropy and perplexity of chord phrases), we examine the generated harmonization from a more complete perspective.

Table 3. Statistics of the Data Set

Albums	*Songs*	*Melody*	*Notes*	*Chords*	*Phrases*
number	6	69	4,739	1,989	396

Experiments and Results

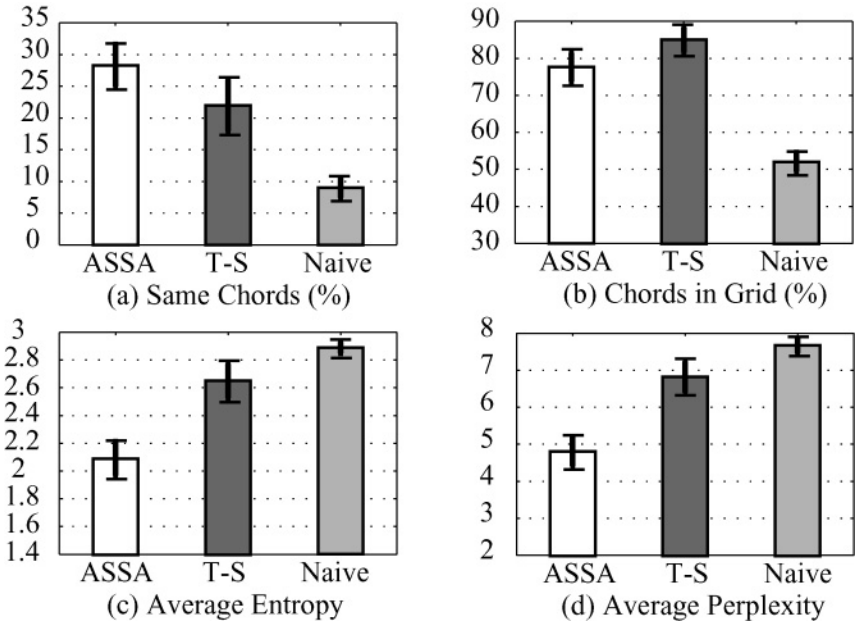
This section describes the three types of experiments we conducted to evaluate the ASSA system using the quantitative metrics defined in the evaluation section. In the inter-system comparisons, we examine the style specificity of the results generated by the ASSA system, the rule-based T-S Harmonic Analyzer (Temperley and Sleator 1999a; Temperley 2002), and a naïve harmonizer with only one constraint. In the intra-system comparisons, we study the impacts of selecting different training songs and of changing the size of the training set. In the case studies, we examine the training songs in more detail to discover the properties that can result in the ASSA system's more style-consistent harmonizations.

Data and Ground Truth

For both the intra- and inter-system experiments, we use the songs from six pop/rock albums by three stylistically distinct bands: Green Day's *Dookie* and *American Idiots*, Keane's *Hopes and Fears* and *Under the Iron Sea*, and Radiohead's *Pablo Honey* and *Hail to the Thief*. Songs from the same album are considered to have similar harmonization styles. For each song, we use the chords given in the published lead sheet as the ground truth. The melody from each musical score is manually encoded as a MIDI file, with a text file containing the 73 attributes describing each melody note. Extended chords on the score are reduced to their basic triads (in order to map them to the neo-Riemannian chord space) which are then saved, along with the melody-chord alignment information, in text files. The harmonization generated by the ASSA system consists of a sequence of letters, where each letter denotes one of the 24 major and minor triads.

Table 3 provides the statistics on the data set for the experiment. Note that for chord-tone determination, the number of melody notes represents the number of instances in the classification task. The number of chords indicates the number of answers that the system needs to generate in each experimental setting. When entropy and perplexity are calculated, the calculation is done at the phrase level. Therefore, the number of phrases provides information about the overall number of samples for the average entropy and perplexity.

Figure 6. Inter-system comparison results: average statistics for the ASSA system, T-S Harmonic Analyzer, and a naïve harmonizer.



Inter-System Comparisons

In this section we compare the ASSA system with two rule-based harmonization systems, the T-S Harmonic Analyzer and a naïve harmonizer.

The T-S Harmonic Analyzer (Temperley and Sleator 1999b) applies preference rules to rhythm analysis and harmonization in the Western classical-music tradition. To harmonize a melody, the system first divides it into segments, then assigns the root of the chords for each segment without indicating the mode (major or minor). For comparison, we select the chord's mode by choosing commonly used chords as described in Kostka and Payne (2003). For example, when a root of G is reported by the Harmonic Analyzer for a segment of a melody in the key of C major, we select G major (the more common option), not G minor, as the chord.

We design a naïve harmonizer as the base case for the comparison. The system adheres to only one constraint: Harmonizing chords must contain at least one of the melodic notes to be harmonized. Chords are randomly chosen from the 24 major and minor triads if no melody appears in the bar.

Figure 6 shows summary statistics comparing the ASSA system, the T-S Harmonic Analyzer, and the naïve harmonizer. The ASSA-generated harmonizations were produced by using all songs except one (from the same album) for training, and created for the held-out song. For each song on the six albums, the ASSA system generated a harmonization for it based on the other songs on the same album. We also used the T-S Harmonic Analyzer and the naïve harmonizer for each song on the six albums. In Figure 6a, the ASSA-generated harmonizations show a higher same-chord average percentage than the other two systems, but more chords in the T-S harmonizations are within the nine-cell chord map range, reported in Figure 6b as the chords-in-grid percentage. This result indicates that ASSA generates more chords that match the original exactly, while more chords generated by the T-S Harmonic Analyzer are closely related to the original. Figures 6c and 6d show the average entropy and average perplexity of the harmonizations generated by the three systems. Note that ASSA produces lower entropy and perplexity values than the T-S analyzer or the naïve harmonizer, which implies that the chord progressions generated by the ASSA system are stylistically more consistent with the original.

In Figure 7 we compare the types of chords generated by the systems. The darkness of the

Figure 7. Inter-system comparison results. Chord map distributions for the ASSA system, T-S Harmonic Analyzer, and a naïve harmonizer. The darkness of the cells on the chord map represents the percentage of generated chords that fall into the corresponding categories, as indicated in the vertical bar on the right.

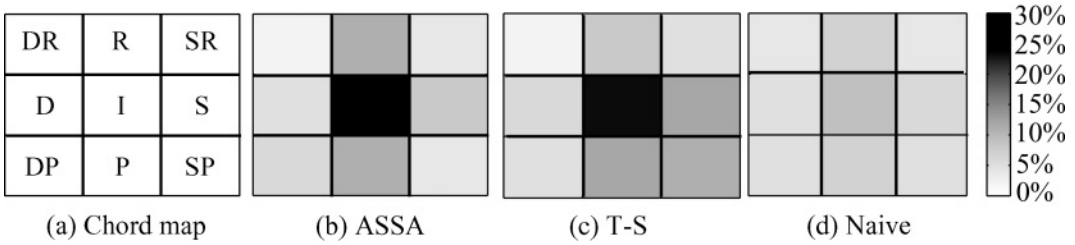


Table 4. Average Rates of Correct Chord-Tone Determination for the Four Training Options

	Single-Best	*Single-Worst*	*All-CTDon*	*All-CTDoff*
CTD correct rate (%)	72.05 ± 1.72	43.62 ± 2.0	62.95 ± 2.57	43.7 ± 2.36

cells on the chord map represents the percentage of generated chords that fall into the corresponding categories, as indicated in the vertical bar on the right in Figure 7. The distribution map in Figure 7b shows that the ASSA system tends to generate chords that are parallel or relative to the original. We observe in Figure 7c that the T-S system generates almost equal numbers of chords in the R, S, and SP boxes. In Figure 7d, we see that the chords generated by the naïve harmonizer are relatively evenly distributed among the close chords.

Intra-System Comparisons

In machine learning tasks, it is usually the case that more training data guarantees better results. To test whether more data is indeed better for ASSA, we designed experiments to compare two kinds of training-set choices. The first selects only one song from an album for training, and tests the model on another song on the same album. The second uses all except one song (from the same album) for training, and tests the model on the held-out song. We apply the two training-set choices to each song in the data set, and compute the average of the individual test results for all the songs on the six albums.

For each song on the six-album data set, we apply the two training-set choices in a total of four ways. In the first case, labeled "Single-Best," the system learns from the one song on the album (other than the held-out song) that has the highest reported rate of correct chord-tone determination. In the second case, labeled "Single-Worst," the system learns from the one song on the album (other than the held-out song) that has the lowest rate of correct chord-tone determination. In the third case, labeled "All-CTDon," the system learns from all songs on the same album (excluding the held-out song). In the last case, labeled "All-CTDoff," the system learns from the training set, as in the third case, but with the chord-tone determination module disabled—i.e., all melody notes are uniformly considered chord tones. The four training options allow us to study the benefit that the chord-tone determination module adds to the neo-Riemannian framework (All-CTDon versus All-CTDoff, and Single-Best versus Single-Worst), as well as the impact of the training set size (Single versus All). Table 4 reports the average rates of correct CTD with their 95% confidence intervals for the four training options. The correct rate serves as an indicator for the accuracy of the information received by the neo-Riemannian framework for generating harmonizations.

We generate harmonizations using each training option. Figure 8 shows the average (a) rate of correct (i.e., same) chord generation and (b) chord-in-grid percentage and their 95% confidence intervals, for the Single-Best, Single-Worst, All-CTDon, and All-CTDoff training data cases. First, note that in Figure 8b all four training sets result in similar chords-in-grid percentages. This result suggests that NROs offer a robust framework for generating chord progressions within an acceptable range, even

Figure 8. Intra-system comparison results using musical metrics. Average statistics for Single-Best, Single-Worst, All-CTDon, and All-CTDoff training data.

Figure 9. Intra-system comparison results using average cross entropy and perplexity. Statistics for Single-Best, Single-Worst, All-CTDon, and All-CTDoff training data.

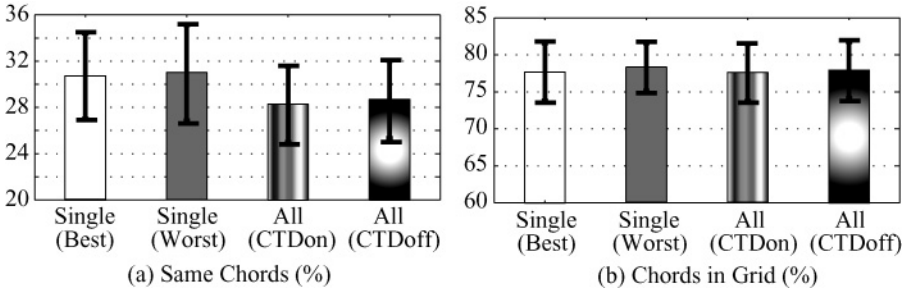


Figure 8.

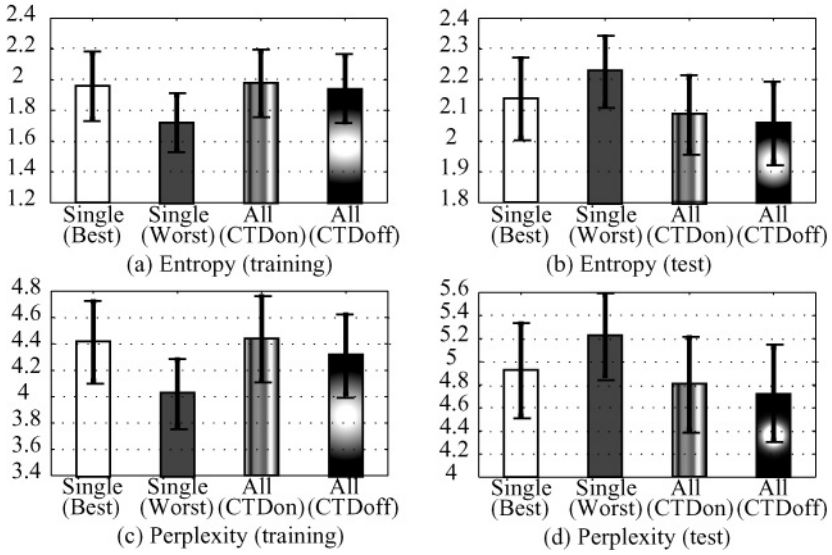


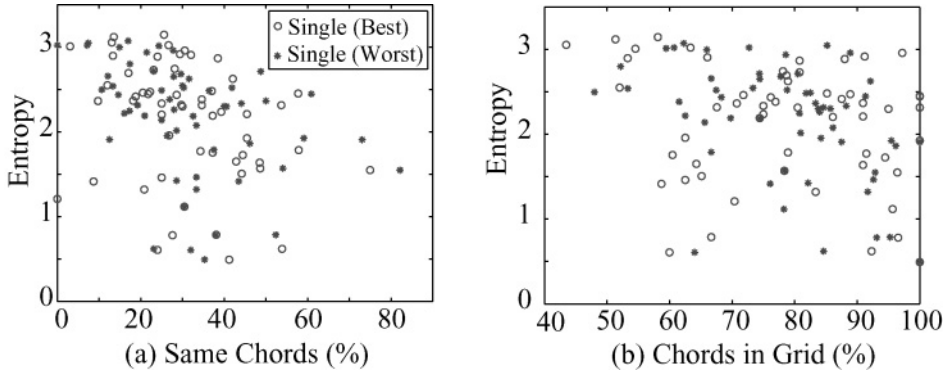
Figure 9.

when chord-tone information is poor. However, in Figure 8a, the system using Single-song training sets (both best and worst case scenarios) reports slightly higher same-chord percentages than the ones with the All-songs training sets. The statistics suggest that for this style emulation model, more training data do not necessarily produce more chords that are correct.

Next, we calculate the cross entropy and perplexity between the generated harmonization and the two data sets, namely, the training data and the test data. For a generated harmonization produced by a model that learned from one of the training set choices, we first calculate the cross entropy and perplexity between it and the training set on which the model is built. We can consider these values as measures of the stylistic difference, in terms of chord progressions in each phrase, between the generated harmonization and that of the songs in the training set. We then calculate the cross entropy and perplexity between the generated harmonization and the test song. These values provide us with a measure of how different the generated chord progressions are from the original (i.e., the ground truth).

Figure 9 shows the average cross entropy and perplexity of ASSA using the four training set

Figure 10. Average entropy between the training song in the data sets (Single-Best and Single-Worst) and the test song (ground truth) versus the corresponding same-chords and chords-in-grid percentages.



options. Observe that the entropy values in Figure 9a are consistently lower than those in Figure 9b, indicating that the chord sequences in the phrases of the generated harmonization contain slightly more patterns in common with the training songs than with the test song. However, the differences between the average entropy and perplexity, for all four training options, are not statistically significant, as shown in the four charts in Figure 9. These similar values imply that the different training strategies have only marginal effects on the ASSA-generated chord progressions. The perplexity values reported can be interpreted as follows: For each of the seven or eight chords that accompany a phrase-length segment in the ASSA-generated harmonization, there are, on average, four to five possible chord choices based on the training and test songs' chord idioms.

Case Studies

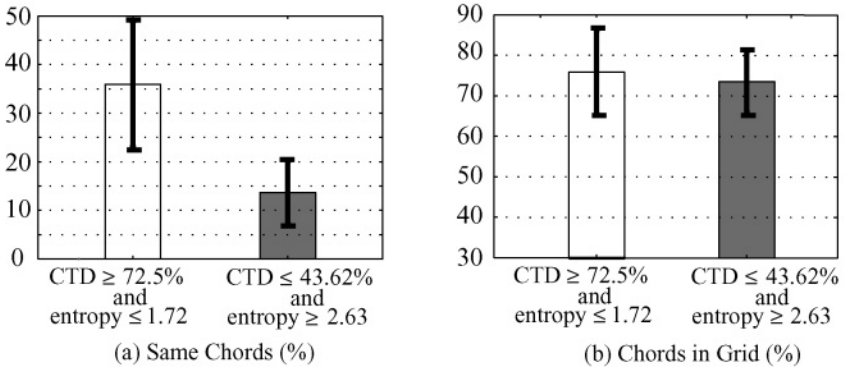
It may seem counterintuitive that the Single-Best training set results in a slightly lower same-chords percentage than does the Single-Worst training set, as shown in Figure 8a. The training example affects the system not only through the chord-tone determination result, it also impacts the transition probabilities between adjacent chords. There might be cases where two songs share similar melody–chord relations in many bars, but contain very different chord progressions. Whereas the chord-tone correct rate describes the melody–chord relation, the average cross entropy between the training and test songs provides information about how dissimilar the chord phrases are between the two. Figure 10 shows a situation where higher chord-tone correct rate (Single-Best) does not guarantee lower entropy, i.e., similar chord progressions.

Therefore, we conclude that the training example that can be used to produce the harmonization with the most similar style must be one that has both a high chord-tone correct rate and a low average cross entropy value.

To determine the degree to which these two factors lead to more style-specific harmonization, and how much the output fluctuates given different training examples, we take two subsets of the training examples. The first subset consists of single training songs that produce chord-tone correct rates above the Single-Best average of 72.5%, and harmonizations having entropy values in the lower quartile (i.e., below 1.72). The songs in the second subset produce chord-tone correct rates below the Single-Worst average of 43.62%, and harmonizations having entropy values in the upper quartile (i.e., above 2.63).

Figure 11 shows the (a) same chords and (b) chords-in-grid percentages for the system using these two training subsets. Note that in Figure 11a, the subset with high chord-tone correct rates and low entropy values clearly outperforms the other one, thus showing that choosing training examples from the same album as the test song can produce a wide range of results. The two training subsets report similar chords-in-grid percentages, however, as shown in Figure 11b, which confirms that the neo-Riemannian framework is robust against poor input examples.

Figure 11. Generating harmonizations using two training subsets. High chord-tone correct rate and low entropy versus low chord-tone correct rate and high entropy.



Conclusions and Discussions

In this article we proposed and demonstrated a hybrid approach to the design and implementation of an automatic style-specific accompaniment system. The system combines a music-theoretic framework with statistical learning to model the harmonization process in a sequence of logical steps, and generates stylistically consistent chord progressions based on only a few examples. The system is designed not only to provide users with easier ways to specify the harmonization style they wish to emulate, but also to serve as a systematic model for the process of generating harmonizations in a variety of different genres.

We proposed quantitative methods for evaluating machine-generated style-specific harmonizations. These methods apply not only to the ASSA system, but also to any system that aims to emulate harmonization style. Summary statistics show the musical relations between corresponding pairs of generated and original chords; entropy and perplexity judge the unpredictability of generated chord sequences with respect to the chord idioms in the original harmonization.

In the inter-system comparisons, we showed that the ASSA system produces more chords identical to those in the original song, and generates harmonizations that are more consistent with the original (having a lower degree of surprise) than the T-S and naïve systems. In the intra-system comparisons, we showed that for the purpose of style emulation, where individual idiosyncrasies in chord choice can be important to the definition of the style, a training set with more examples, even if all songs are from the same album, does not guarantee better results. We also demonstrated that neo-Riemannian theory combined with probability distribution smoothing provides a robust framework for generating reasonable chord progressions in the event of poor chord-tone information. In the case studies, we observed that the quality of ASSA-generated harmonizations varies when different single training songs, even those from the same album, are used. The best result is attained when the training song choice results in a high chord-tone correct rate and a harmonization that has low entropy with respect to the test song.

In addition to the system we proposed for generating style-emulating harmonizations, it is also our goal to encourage further discussions on two related topics. The first considers the way in which such systems are evaluated. As mentioned in the "Related Work" section, most researchers use participants' subjective opinions to evaluate system-generated harmonizations. Human evaluation may be a solution for tasks such as music recommendation and genre classification; it can be prone to many personal biases when judging computer-generated chords that emulate musicians' chord choices, however. We have been conducting Turing tests whenever the system is presented in conferences or at invited talks. Although respondents found the system-generated harmonization and the original to be indistinguishable, we did not keep a formal count of the results of such tests. We found that it is

difficult for professional musicians to only concentrate on the harmonization style of the composer without being affected by their own harmonization preferences. In general, people make judgments about music (who is playing what piece by whom) based on split-second exposure to musical timbre (Krumhansl 2010), a time scale too small in which to consider chord transitions. Although the absence of timbre and vocal cues is desirable for comparing chord choices, it is difficult for people to ignore their associative memory and simply judge a musician's style based on these choices. Thus, we argue that for evaluating systems that emulate a particular harmonization style, entropy and perplexity provide an objective and unbiased tool, just as they do in natural language processing. In our future work, we plan to collaborate with composers to ask them to judge the quality of generated harmonizations when the system is trained to emulate their style, based on their own compositions, and see whether their ratings correlate well with the results computed using entropy and perplexity.

The second topic for further discussion relates to the effectiveness of rule-based systems versus data-driven ones for style emulation. It may be widely assumed that learning systems are more suitable than rule-based ones for such a task. However, there is evidence in the literature that a rule-based system can outperform pure statistical approaches for style-emulating harmonization. For example, Phon-Amnuaisuk and Wiggins (1999, p. 28) compared rule-based and data driven (learning) approaches to the four-part harmonization problem and showed in their study that the "KBS (knowledge based system) is better suited for this [harmonization] task." Furthermore, by training a system on hundreds of examples, as is done in MySong (Morris, Simon, and Basu 2008), general chord patterns prevail, which effectively leads to the kinds of chord patterns described in rule-based systems. In the future, we will compare the ASSA system with a learning-oriented system, such as one using hidden Markov models, to examine the manner in which the size of the training set affects the uniqueness of the harmonization style.

Finally, we plan to make the ASSA system more accessible to amateur composers as a composition software tool by adding an interactive user interface (see Chuan and Chew 2010 for a preliminary design). Another promising application of the ASSA model and the style similarity assessment metrics is the generation of music database indices for music information retrieval.

Acknowledgments

This work was supported in part by National Science Foundation (NSF) Grant No. 0347988. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors, and do not necessarily reflect those of the NSF.

References

Allan, M., and C. K. I. Williams. 2004. "Harmonising Chorales by Probabilistic Inference." In *Proceedings of Advances in Neural Information Processing Systems,* vol. 17, pp. 25–32.

Capuzzo, G. 2004. "Neo-Riemannian Theory and the Analysis of Pop-Rock Music." *Music Theory Spectrum* 26(2):177–199.

Chew, E., A. Volk, and C.-Y. Lee. 2005. "Dance Music Classification Using Inner Metrical Analysis: A Computational Approach and Case Study Using 101 Latin American Dances and National Anthems." In *Proceedings of the 9th INFORMS Computing Society Conference,* pp. 355–370.

Childs, A. 1998. "Moving Beyond Neo-Riemannian Triads: Exploring a Transformational Model for Seventh Chords." *Journal of Music Theory* 42(2):181–193.

Chuan, C.-H., and E. Chew. 2007. "A Hybrid System for Automatic Generation of Style-Specific Accompaniment." In *Proceedings of the 4th International Joint Workshop on Computational Creativity*, pp. 57–64.

Chuan, C.-H., and E. Chew. 2008. "Evaluating and Visualizing Effectiveness of Style Emulation in Musical Accompaniment." In *Proceedings of the 9th International Conference on Music Information Retrieval,* pp. 57–62.

Chuan, C.-H., and E. Chew. 2010. "Quantifying the Benefits of Using an Interactive Decision Support Tool for Creating Musical Accompaniment in a Particular Style." In *Proceedings of the 11th International Conference on Music Information Retrieval*, pp. 471–476.

Cohn, R. 1996. "Maximally Smooth Cycles, Hexatonic Systems, and the Analysis of Late-Romantic Triadic Progressions." *Music Analysis* 15(1):9–40.
Cohn, R. 1997. "Neo-Riemannian Operations, Parsimonious Trichords, and their Tonnetz Representations." *Journal of Music Theory* 41(1):1–66.
Farbood, M., and B. Schoner. 2001. "Analysis and Synthesis of Palestrina-Style Counterpoint Using Markov Chains." In *Proceedings of the International Computer Music Conference* (pages unnumbered).
Hook, J. 2002. "Uniform Triadic Transformations." *Journal of Music Theory* 46(1/2):57–126.
Huang, C.-Z., and E. Chew. 2005. "Palestrina Pal: A Grammar Checker for Music Compositions in the Style of Palestrina." In *Proceedings of the 5th Conference on Understanding and Creating Music* (pages unnumbered).
Kochavi, J. 2002. *Contextually Defined Musical Transformations*. PhD Dissertation, Music Department, State University of New York at Buffalo.
Kostka, S., and D. Payne. 2003. *Tonal Harmony*. New York: McGraw-Hill.
Krumhansl, C. L. 2010. "Plink: 'Thin Slices' of Music." *Music Perception* 27(5):337–354.
Lee, H. R., and J. S. Jang. 2004. "i-Ring: A System for Humming Transcription and Chord Generation." In *Proceedings of IEEE International Conference on Multimedia and Expo*, pp. 1031–1034.
Lewin, D. 1987. *Generalized Musical Intervals and Transformations*. New Haven, Connecticut: Yale University Press.
Lidstone, G. 1920. "Note of the General Case of the Bayes-Laplace formula for Inductive or a Posteriori Probabilities." *Transactions of the Faculty of Actuaries* 8:182–192.
Manning, C. D., and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: MIT Press.
Mitchell, T. 1997. *Machine Learning*. New York: McGraw-Hill.
Morris, D., I. Simon, and S. Basu. 2008. "MySong: Automatic Accompaniment Generation for Vocal Melodies." In *Proceedings of Computer-Human Interaction*, pp. 725–734.
Phon-Amnuaisuk, S., A. Tuwson, and G. Wiggins. 1999. "Evolving Music Harmonisation." In *Proceedings of Fourth International Conference on Neural Networks and Genetic Algorithms*, pp. 229–234.
Phon-Amnuaisuk, S., and G. Wiggins. 1999. "The Four-Part Harmonisation Problem: A Comparison between Genetic Algorithms and a Rule-Based System." In *Proceedings of the Society for the Study of Artificial Intelligence and Simulation of Behaviour*, pp. 28–34.
Ponsford, D., G. Wiggins, and C. Mellish. 1998. "Statistical Learning of Harmonic Movement." *Journal of New Music Research* 28(2):150–177.
Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. San Mateo, California: Morgan Kaufmann.
Stephenson, K. 2002. *What to Listen for in Rock: A Stylistic Analysis*. New Haven, Connecticut: Yale University Press.
Temperley, D. 2002. *The Cognition of Basic Musical Structures*. Cambridge, Massachusetts: MIT Press.
Temperley, D., and D. Sleator. 1999a. "Modeling Meter and Harmony: A Preference Rule Approach." *Computer Music Journal* 15(1):10–27.
Temperley, D., and D. Sleator. 1999b. "Harmonic Analyzer." Available on-line at www.cs.cmu.edu/~sleator/harmonic-analysis. Accessed July 2011.
Volk, A. 2002. "A Model of Metric Coherence." In *Proceedings of the 2nd Conference on Understanding and Creating Music* (pages unnumbered).
Witten, I. H., and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco, California: Morgan Kaufmann.