LyricAlly: Automatic Synchronization of Textual Lyrics to Acoustic Music Signals

Min-Yen Kan, Ye Wang, Denny Iskandar, Tin Lay Nwe, and Arun Shenoy

***Abstract*—We present LyricAlly, a prototype that automatically aligns acoustic musical signals with their corresponding textual lyrics, in a manner similar to manually-aligned karaoke. We tackle this problem based on a multimodal approach, using an appropriate pairing of audio and text processing to create the resulting prototype. LyricAlly's acoustic signal processing uses standard audio features but constrained and informed by the musical nature of the signal. The resulting detected hierarchical rhythm structure is utilized in singing voice detection and chorus detection to produce results of higher accuracy and lower computational costs than their respective baselines. Text processing is employed to approximate the length of the sung passages from the lyrics. Results show an average error of less than one bar for per-line alignment of the lyrics on a test bed of 20 songs (sampled from CD audio and carefully selected for variety). We perform a comprehensive set of system-wide and per-component tests and discuss their results. We conclude by outlining steps for further development.**

***Index Terms*—Acoustic signal detection, acoustic signal processing, music, text processing.**

I. Introduction

THANKS to lowered costs of storage, different media are often used to record and represent a single event. News reports are captured as MPEG video, WAV audio, and with closed caption transcripts; popular music is often encoded as MP3 and also represented by lyrics. When multiple forms of media documenting the same object are available, the problem of alignment arises.

This paper presents a solution and implemented system for one such alignment problem: automatically synchronizing music to its lyrics. More formally, given an acoustic musical signal with sung vocals and a transcription of the vocals (its lyrics) as a set of lines, we seek the time offset and duration of each lyric line. We developed our system, LyricAlly, with popular music in mind to maximize its real-world utility, and during development found that the regular structure of popular music greatly helps to constrain the problem and increase performance.

Once lyrics are synchronized to a song, random access is greatly facilitated. When paired with an indexing and search interface, a user can easily browse and listen to different lines of a song that contain specific lyrics across a music library. Such line alignment allows users to zoom quickly to vocal sections, skipping nonvocal parts. LyricAlly also facilitates nonvocal music access indirectly by labeling music sections (i.e., intro, chorus, and verse), such that the user can jump to any instrumental section between vocal sections. Such synchronization also facilitates karaoke, where the user sings along with the song. Today, karaoke lyric timestamps are manually input employing tedious human effort.

To solve the synchronization problem, we use a multimodal approach. A key finding in our work is that the appropriate pairing of audio and text processing on the separate modalities helps to create a more accurate and capable system. The modalities complement each other by providing information that is not easily obtainable in the other. Our use of music knowledge reduces the complexity of downstream tasks and improves system performance. We decompose the problem as two separate subtasks performed in series: a high-level song section (e.g., chorus, verse) alignment subtask, followed by a lower level line alignment subtask. In comparison to a single-level alignment model, we feel that this cascaded architecture boosts our system's performance and allows us to pinpoint components responsible for performance bottlenecks.

LyricAlly is a complex system whose development has been documented in several previous publications [10], [21], [27]. In this paper, we bring together these works to highlight the audio processing components. Additionally, we extend our previous work to give new, detailed, per-component performances and error analyses. We have also clarified and expanded on how our rule-based integration functions.

In Section II, we give an overview of related work. We then introduce LyricAlly in Section III and give definition of terms used throughout the paper. A description of the innovations and details of its audio and text components follows. We then detail our two-level integration of components in Section VI. We analyze the system's performance and conclude with comments on current and future work.

II. Related Work

Managing and coordinating multimedia is a growing area of research as more data requiring such processing becomes available. However, to our knowledge, little previous work addresses the problem of automatically synchronizing lyrics to music signals. Clearly, interest is present, as [7] describes a framework to embed lyric timestamps inside music files but does not describe how to obtain them. In [32], Zhu *et al.* also consider the processing of karaoke media in which video and audio streams are already synchronized. Both works focus on utilizing embedded alignment metadata, rather than automatically computing it as

Manuscript received December 6, 2006; revised October 3, 2007. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Masataka Goto.

M.-Y. Kan, Y. Wang, and A. Shenoy are with the School of Computing, National University of Singapore, Singapore 117543.

D. Iskandar and T. L. Nwe are with the Institute for Infocomm Research, Singapore 119613.

Digital Object Identifier 10.1109/TASL.2007.911559

in our case. The Semantic Hi-Fi project [6] is also pursuing similar goals of automatic synchronization as LyricAlly, but thus far has not reported any results of their work. In the rest of this section, we compare other alignment systems to LyricAlly; works related to individual components are discussed later in the respective component sections.

In the speech recognition community, spoken document retrieval attempts to align potentially inaccurate closed-caption transcripts of news reportage to the corresponding video [4], [18], [24], [25], [31]. Typically, a video corresponds to an entire hour-long newscast, and coarse-grained alignments where alignment can be off by more than a minute are acceptable, as a video clip with the context is retrieved, rather than the exact offset of a relevant word or phrase. Clearly, more fine-grained accuracy is needed for karaoke applications where alignments off by a single second can be unsatisfactory. Our application domain is also more challenging in the sense that music has a lower signal-to-noise ratio, due to distortion from instrumental music accompaniment.

In [26], the authors use a large-vocabulary speech recognizer to perform lyric recognition, but their system only works with pure singing voice. However, our previous work [27] shows that real-world acoustic musical signals are significantly more complex than pure singing; models created for pure singing voice require considerable adaptation if they are to be successfully applied to acoustic musical signals. Our experience shows that the transcription of lyrics from polyphonic songs using speech recognition is an extremely challenging task. This has been validated by other groups [12]. This difficulty has led us to reexamine the transcription problem. We recognize that transcription is often not necessary, as many lyrics are already freely available on the Internet. As such, we formulate the problem as one of lyric alignment rather than transcription. This design decision implies that traditional speech recognition is not employed in LyricAlly—we attempt to detect the presence of singing voice but not recognize its contents.

In a sense, our work is analogous to those in [1], [5], and [22] which perform automatic audio-MIDI alignment. However, their task is quite different from ours, as MIDI files provide both timing and frequency information while lyrics normally do not. In audio-MIDI alignment, it has become fairly standard to use dynamic time warping to align polyphonic music to scores [5], [20]. A key difference is that in MIDI-audio synchronization, one can transform audio data from both MIDI and audio sources into a common, feature-based representation (peak structure distance in [20], chromagrams in [5]), whereas in lyrics-audio synchronization one only has such information from the original polyphonic source.

III. System Description

LyricAlly's architectural decisions are mainly motivated by the principle of using music knowledge to constrain and simplify the synchronization process whenever possible. As we deal with music rather than general audio, we first process the acoustic signal to find the tempo and segment the signal into discrete, musical bar-sensitive time units for further processing. LyricAlly features a two-phase architecture in which both audio and text analysis are first performed independently. In the second phase, LyricAlly then aligns these two modalities in a top-down approach to yield the final synchronization results, as shown in Fig. 1.

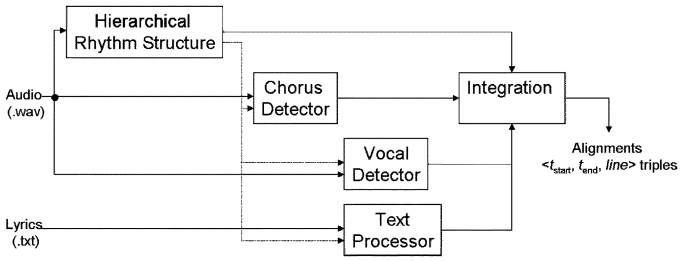

Fig. 1. Block diagram of our LyricAlly lyric alignment system.

We pause to give definitions of the structural elements of music (also referred to as sections) used in our paper.

Intro: The opening section that begins the song, which does not have singing voice, may contain silence and may lack a strong beat (i.e., it may be arrhythmic).

Verse: A section that roughly corresponds with a poetic stanza and is the preamble to a chorus section.

Chorus: A refrain section. It often sharply contrasts the verse melodically, rhythmically, and harmonically, and assumes a higher level of dynamics and activity, often with added instrumentation. As noted in [9], chorus sections are often not identical, but may vary in sung lyrics and with added or inserted repetition.

Bridge: A short section of music played between the parts of a song. It is a form of alternative verse that often modulates to a different key or introduces a new chord progression.

Outro (or Coda): A section that brings the song to a conclusion. For our purpose, an outro is a section that follows the bridge until the end of the song. This can be characterized by the repetition and fading out of the chorus.

Music exhibits a wide range of melodic and rhythmic characteristics. LyricAlly has been built and evaluated with real-world popular music, and as such, to make the problem tractable and simplify our work, we limit the scope of music that LyricAlly handles. First, the songs must have a common time signature (i.e., 4/4; four beats to a bar). Second, the structure of song is rigid: comprising an intro, two verses, two choruses, a bridge, and an outro. Finally, the tempo of the song should roughly be constant and in the range of 40–185 beats per minute. Thus, we limit the scope of our dataset to music with a strong percussive element. In our informal survey of popular songs, these restrictions are not overly restrictive—over 40% of songs surveyed fit this profile. With respect to the lyrics, we assume that each section's lyrics are marked as a single block of lines, and that the lyrics accurately reflect the complete set of words sung in the song. It should be noted that accurate lyrics can be found utilizing the Web to find and align multiple lyrics [11]. Finally, note that certain components do not have these restrictions; the requirements above are strictly the union of all limitations of each component.

As we detail the workings of the components in Sections IV–VII, we will use a running example of such a song, *25 Minutes*, performed by Michael Learns To Rock (MLTR).

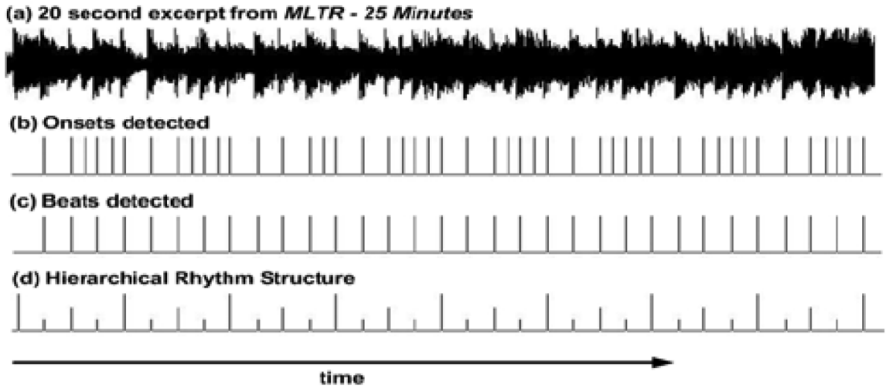

Fig. 2. Rhythm structure of an excerpt in *25 Minutes*.

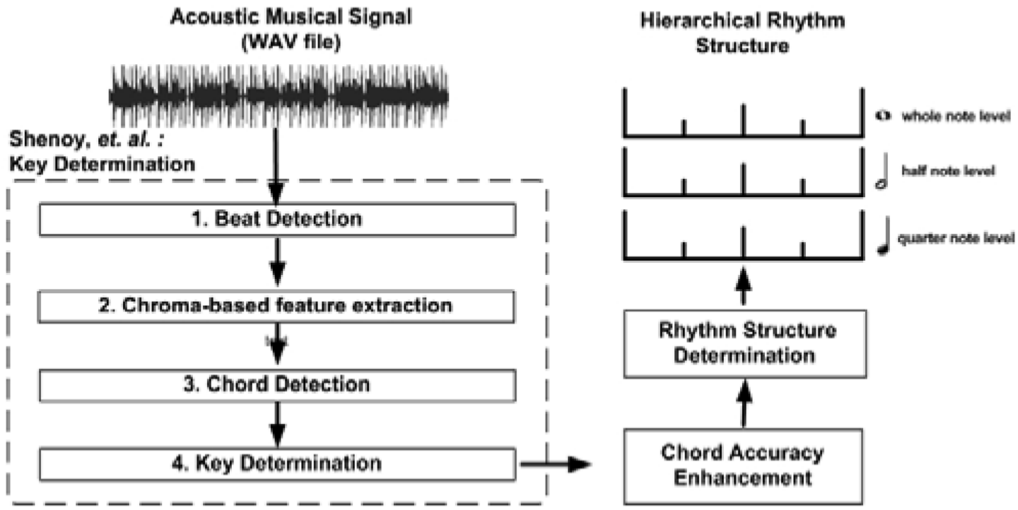

Fig. 3. Block diagram of our hierarchical rhythm structure component.

IV. Audio Processing

Audio processing in LyricAlly attempts to localize sections of vocals, which are then refined to line level alignment in a final integration component. In technical terms, this processing has two objectives: finding the boundaries of repeated chorus sections and finding time spans in the input music that contains singing vocals. Each objective is achieved by a separate component as illustrated in Fig. 1. Note that even though the two components' objective may overlap, this is not necessarily the case: repeated sections may not necessarily have singing voice.

The raw acoustic music signal is voluminous, and processing it using short 80-ms frames as was done in [9] would be very time consuming. As we are dealing with music, we can reduce the search space by using a longer time unit motivated by music knowledge. Instead of using short time frames as base time units, our work uses a frame width equal to the interbeat interval (IBI), which is often significantly longer. Processing is also assisted when the hierarchical rhythm structure is known. We first discuss how this preprocessing step is performed, then discuss chorus and vocal detection.

A. Hierarchical Rhythm Preprocessing

Hierarchical rhythm processing finds both the tempo of the input song and its bar structure. In the common time signature, the latter step determines the strong beats (the beats that fall on the 1 and 3 beats of the bar) and the weak beats (the beats that fall on the 2 and 4 beats of the bar). Beat detection is done to first find the tempo of the music and further analysis is done to derive the rhythm structure.

1) Beat Detection: Beat detection in LyricAlly follows standard beat processing: we find salient onsets in subbands of the input using an energy-based function with adaptive thresholding [28]. Onset detection results from all subbands are then combined to produce a series of detected onsets, as shown in line (b) of Fig. 2. To find the tempo, LyricAlly builds a histogram of the intervals between any two onsets (not necessarily consecutive) for the entire song. We also identify relationships with other intervals to recognize harmonic relationships between the beat (quarter note level) and simple integer multiples of the beat (half and whole notes). Intervals allow for a margin of 25 ms accounting for slight variations in the tempo. The interval length that has the highest frequency is assumed to be the interbeat interval (IBI), equivalent to a quarter note, given that the input is in common time. We then track only the onsets that are separated by exactly the IBI and interpolate any missing beats, as shown in the line (c) of Fig. 2.

2) Finding Hierarchical Structure: The architecture of our hierarchical rhythm detector is summarized in Fig. 3, which fur-

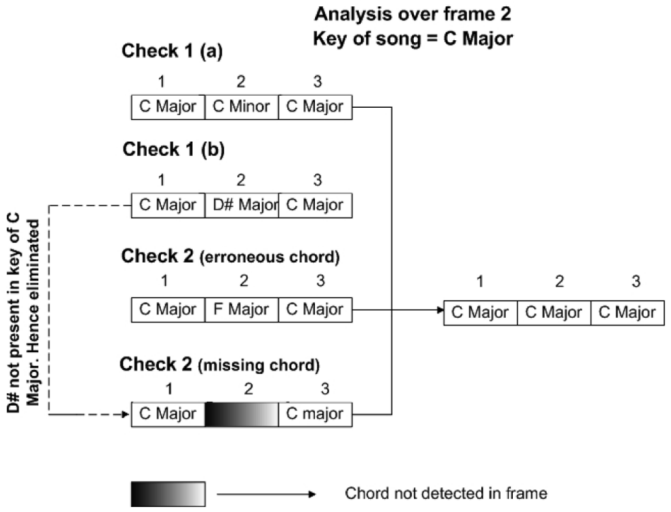

Fig. 4. Chord correction. Four different scenarios where three consecutive IBI chord detection results are corrected. Check 1 checks for chords not in the song's key, then replaces it with one in the same pitch class (row 1; C Minor → C Major) or eliminating it altogether (row 2; D# Major → missing). Check 2 coerces chords in adjacent beats to conform, correcting both erroneous (row 3) and missing chord (row 4) errors.

ther builds on beat detection to derive the hierarchical rhythm structure given in line (d) of Fig. 2.

Previous work [8] states that chord changes in music are more likely to occur at the beginning of a bar than at other beat positions. Thus, if we can detect chord changes, we can use this information to infer the hierarchical rhythm structure.

Chord detection is done by calculating chroma vectors for each beat (IBI). By formulation, a chroma vector captures the 12 pitch classes (each class is associated to a semitone) across multiple octaves. Chroma vectors are intended to capture the harmonic characteristics and be robust to different instrumentations. We use the chroma vectors to compute the chords present in each beat by extracting three prominent frequencies and comparing them against a set of templates that describe all possible chord types with three notes (triads).

Chord detection accuracy can be further improved using knowledge of the song's key. We assume that songs use a single key throughout the entire song. According to music theory, there are 12 major keys and 12 minor keys. Each key has a unique combination of chords. The key detection computes cosine similarity measure between all the detected chords and the chords in each of the 12 major/minor keys. The key detection returns the key that is most consistent with the histogram of the detected chords [21]. Using the derived key, we can filter out incorrectly detected chords that are inconsistent with the key, increasing the accuracy of the chord detector from 48.1% (without key filtering) to 63.2% (with filtering). As highlighted earlier, chord changes in music are more likely to occur at the beginning of the bar (strong beat 1). Thus, within the bar, neighboring chords are most often identical. We coerce chords at consecutive IBIs to be consistent with one another, to smooth and filter the chord detection results, as shown in Fig. 4. For beat i, we look at the detected chords in beat $i-1$ and beat $i+1$. If the detected chords in these adjacent beats are the same, we assign the chord at beat i to be same chord as its neighbors.

The bar boundaries are then identified as follows: Every beat location could be a possible measure boundary. This set is further refined by identifying only those beat locations that have the same chord in four consecutive beat boundaries. This follows from our premise that chord changes are likely to occur at the beginning of the bar. However inaccuracies in chord detection may lead to incorrect detection of bar boundaries. We thus compute all possible patterns of these beats with spacing in multiples of four. The longest pattern is selected as the bar boundary and all the missing bar boundaries are interpolated. In prior work [21], it has been highlighted that the beat detection technique used does not perform very well for drumless music signals since the onset detector has been optimized to detect the onset of percussive events. Additionally, the system interpolates missing beats, thus assuming that there is no break in the inter-beat-interval structure throughout the length of the song. This assumption would however not hold in cases where there is a rest in the music which does not correspond to exactly a few bars. This can often be seen in expressive performances. Thus, for the dataset used, the accuracy of the beat detection is a little less than 90%. Failure of the beat detection would have a direct impact on the accuracy of the rhythm detection. The rhythm detection has, however, been observed to be more robust to low chord detection accuracy, achieving a net accuracy of around 84%.

B. Chorus Detection

LyricAlly implements a repetition based chorus detection algorithm adapted from previous work by Goto [9]. This method defines a chorus as a section in the input that is most often repeated. In practice, determining the chorus section is difficult as choruses are not strictly identical and the length of the chorus is not known. To counter the first difficulty, chroma vectors are used as the representation for repetition detection as they are largely robust to variations caused by using different instruments to carry the same melody. The added difficulty of key modulation in choruses is also addressed by using chroma vectors—by looking for shifts of the same chroma patterns up or down.

A key difference in our work is in computing chroma vectors only per beat rather than per frame. In the original method specified by Goto, 4096-sample frames are used as time units with 1280-sample shifts between two consecutive frames (around 30-ms frames). In contrast, we simplify the representation using the IBI as the time unit. On average this calculates over 90% fewer chroma vectors than the original algorithm, while maintaining similar accuracy. Since chorus detection does exhaustive pairwise comparison, its complexity is $O(n^2)$, resulting in a total savings of around 98% over the original algorithm. Fig. 5 illustrates automatic detection results on *25 Minutes*, in which the two choruses and outro (the long, textured box at the end) have been detected and labeled.

As the focus of this component's implementation centered on scalability, we have yet to evaluate this component with respect to accuracy on the dataset. However, as it performs identically to earlier work by Goto, we assume that our component's accuracy is similar to the 80% that he reported [9].

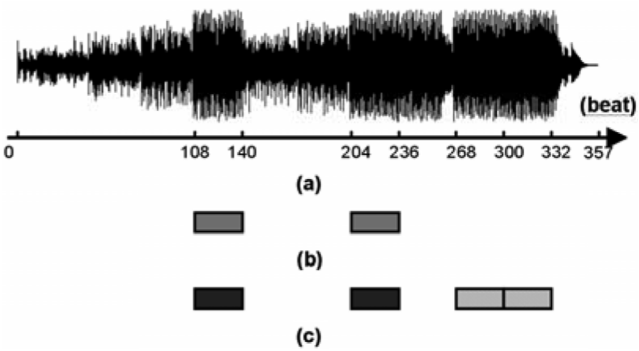

Fig. 5. (a) Waveform of the song *25 Minutes*. (b) Manually annotated chorus sections. (c) Automatically detected chorus sections.

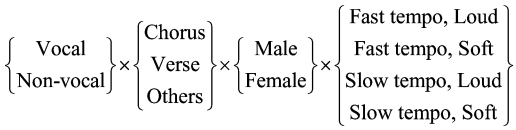

Fig. 6. Multimodel HMM configuration.

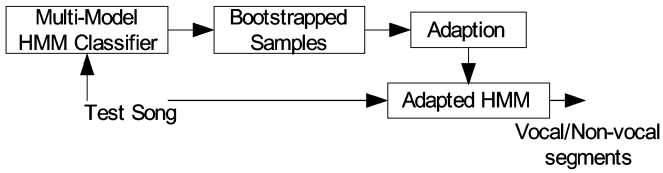

Fig. 7. Block diagram of the vocal detection component, featuring adaptation and segmentation processes.

C. Vocal Detection

LyricAlly equates finding vocal segments with classifying each time unit of the song (IBIs) as either vocal or nonvocal. Similar to much work in speech recognition, we employ standard hidden Markov models (HMMs) for classification [2], [3], [23]. For example, a basic approach employs two HMMs, one each for the vocal and nonvocal classes. Given an input, the model with the lower perplexity is considered to have generated the input.

In contrast to this conventional HMM training method, we create multiple HMM models (referred to as multimodel HMM or MM-HMM) to represent the vocal and nonvocal classes, as we feel a single model for each class does not accurately reflect the ground truth. For example, it is immediately obvious that the spectral characteristics of pure vocals and vocals with instruments are different, although both should be classified as having vocals.

We extract features based on the energy distribution in different frequency bands [19] to differentiate vocal from nonvocal segments, using the interbeat interval as the unit time frame. Finally, we also employ model adaptation which tunes the baseline HMMs to the specific input song, further increasing accuracy.

1) Multimodel HMM for Vocal Segment Classification: We use a MM-HMM framework specifically to capture the intrasong and intersong signal variations. Section types often can account for a signal's intrasong variations. For example, signal strength usually varies among the different section types in popular songs. From our observations, the signal strength of intros is lower than that of verses and choruses. Choruses are usually stronger in comparison with verses since they may have more complex drums, additional percussion, fuller string arrangement, or an additional melody line [29]. Bridges usually have lighter arrangement than choruses and have stronger arrangement than verses. Outros may repeat vocal phrases from the chorus. Intersong signal variations we feel are harder to quantify. In LyricAlly, we introduce the usage of tempo and loudness features to model intersong variation, which we have observed as good distinguishing features on our training dataset.

Our MM-HMM vocal and nonvocal classes are thus characterized as separate models that differ with respect to these dimensions: section type, male/female vocals, and tempo/loudness. The training data (vocal or nonvocal segments) is manually labeled based on the section type (three types: chorus, verse, or other), the sex of the singer (two types), and classified with respect to tempo and loudness (four types). In this way, we build $3 \times 2 \times 4 = 24$ HMMs for the vocal class and 24 HMMs for the nonvocal class. This framework is illustrated in Fig. 6. Each of the 48 HMMs is trained using standard subband-based log frequency power coefficients (LFPCs) as feature vectors. We use an ergodic HMM topology, using forward–backward to estimate transition parameters for each of the vocal/nonvocal models.

TABLE I
AVERAGE VOCAL/NONVOCAL MISCLASSIFICATION ERROR RATES

	SVM	Basic HMM	MM-HMM	Adapted HMM
Error Rate	30.6%	28.2%	25.9%	24.8%

2) Adapting HMMs Using Bootstrapping: Tempo, timbre, and loudness are the song specific characteristics. These characteristics differ among songs. If we can create a vocal detector that learns the characteristics of the vocals and instruments of a particular song, we believe higher vocal detection accuracy can be achieved. We propose an adaptation method that tailors the baseline models towards song-specific vocal and nonvocal characteristics, inspired by [23]. The original work in [23] requires human annotated vocal and nonvocal segments (bootstrapped samples) of every test song to build song-specific vocal and nonvocal models and also requires having a number of samples for adaptation.

To circumvent the need of human annotation, we explore a three-step semiautomatic approach, as shown in Fig. 7.

- Segment a test song into vocals and nonvocals IBIs (bootstrapped samples) using MM-HMM training method.
- Use these bootstrapped samples to build a song-specific vocal and nonvocal models (adapted HMM models) for each individual test song. The adaptation process modifies a basic two-class HMM classifier.
- Finally, perform a second round of vocal detection using the adapted HMM.

This adaptation process is similar to the unsupervised acoustic segment modeling approach [13] in speech recognition.

3) Evaluation: We conducted a comparative experiment using the basic HMM, MM-HMM, and adapted HMM. We also compared the performance of our HMM-based methods against a SVM-based classification method using the same audio features.

The average vocal/nonvocal misclassification error rates are presented in Table I which shows that the basic HMM which incorporates contextual information reduces error over the baseline SVM classifier by 2.4%. Within HMM-based approaches,

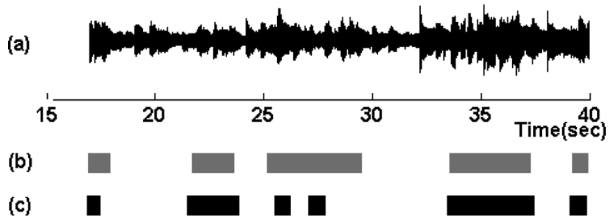

Fig. 8. Vocal detection results. (a) The segment (Verse 1, 23 s) of *25 Minutes*. (b) Manually annotated vocal segments. (c) Automatically detected vocal segments.

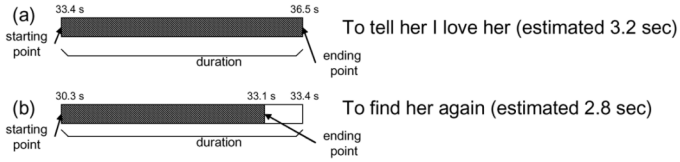

Fig. 9. Finding ending offsets in *25 Minutes*. (a) Case where the bar length overrides the initial estimate. (b) Case in which the initial estimate is used.

TABLE II
ERROR RATES FOR THE TEXT-BASED DURATION ESTIMATION COMPONENT

	Per Section (n =180)	Per Line (n = 1438)
Error Rate	**37%**	**51%**

the MM-HMM outperforms the basic HMM training approach, while adaptation further reduces error rate. Note that error reduction of the adapted HMM over the MM-HMM is not significant, but that the combined MM-HMM and model adaptation reduce vocal/nonvocal classification error significantly over the basic approach (28.2% to 24.8%). Fig. 8 shows a sample result of manually and automatically detected vocal segments.

We have examined the source of errors in our vocal detection component. In songs where the background instrumental sound dominates vocals, detection error rates increase. Also, we note that the adaptation process improves performance only when the basic HMM performs relatively well. The fact that adaptation improves performance gives evidence that our strategic application of vocal detection is quite good.

V. Lyric Processing

Input song lyrics are processed by a text analysis component that estimates the duration of the lyrics, first producing a rough estimate for each section, then refines these to line-level durations using tempo information derived from hierarchical rhythm preprocessing. The text processor makes duration estimations based on supervised training. In training, we learn durations of phonemes in singing voice. In testing, we decompose lyrics into phonemes and use the learned phoneme durations to calculate a duration estimate.

We estimate phoneme length from singing training data. As phoneme durations in song and speech differ, we do not use information from speech recognizers or synthesizers. Ideally, we should measure the lengths of sung phonemes directly, but gathering a large set of such measurements is tedious and difficult, as individual phoneme durations are short. We choose instead to manually annotate a large dataset at a coarser grained level and interpolate individual phoneme durations. In our scheme, human annotators delimit the start and end times for entire lines of sung lyrics. These annotations effectively give the duration of each line of song. We use dictionary lookup in the CMU phoneme dictionary [30] to map the words in each sung line to an inventory of 37 phonemes. We uniformly parcel the duration of a line among the phonemes which make up the line. After processing many lyric lines in this manner, we acquire a set of different duration estimates for each of the 37 phonemes. For simplicity, we view this set of estimates as observed samples of an underlying Gaussian distribution. We thus calculate the mean and variance of the samples to use in testing.

In testing, the component first estimates the duration for each section of a test song and later refines the estimates to each section's individual lyric lines. It first decomposes a section to its component inventory of phonemes by the same dictionary lookup process, as done in training. The estimate is the sum of the mean duration estimates for the individual phonemes in the section. Note that our approximations model the section as a set of phonemes being sung in sequence without pauses.

We use the same process to derive initial estimates for lines based on their phoneme inventory. Once the tempo of the song is known from the rhythm analysis, we refine these durations using music knowledge. As the input is a song in common time, we assume that lines should have a duration that is an integer multiple of 1/2, 1, 2, or 4 bars. We thus round the initial estimates to an integer multiple of the bar. We then calculate the majority bars per line for each song, and force other line durations within the song to be either 1/2 or 2 times this value. For example, songs in which most lines take 2 bars of time may have some lines that correspond to 1 or 4 bars.

The text model developed thus far assumes that lyrics are sung from the beginning of the bar until the end of the bar, as shown in line (a) of Fig. 9. When lyrics are actually shorter than the bar length, there is often a pause in which vocals rest to breathe before singing the subsequent line, as shown in line (b) of Fig. 9. To model this correctly, we first calculate the ending time for each line by summing up constituent phoneme durations as described above. Then, we check the following conditions. For lines that are estimated to be shorter and were rounded up by the rounding process, vocals are assumed to rest before the start of the following line. In these cases, the initial estimate is used as the ending offset. For lines that are initially estimated to be too long and were rounded down in the process, we predict that the singing leads from one line directly into the next, and that the ending offset is the same as the duration. The ending point of the line is thus given by starting point plus ending offset.

Table II shows the error rates of the duration subsystem for both sections and lines, in tenfold cross validation on our dataset. The line estimates given are after the tempo based correction. Any duration within two standard deviations ($\sim 95\%$ confidence interval) of the actual duration is deemed as correct.

VI. System Integration

With the component results calculated, we proceed to integrate the evidence from the audio and text components. Note that in our current implementation, LyricAlly only aligns the main sections of the song: namely, the two verses and the two choruses; vocals in the bridge and outro are not aligned.

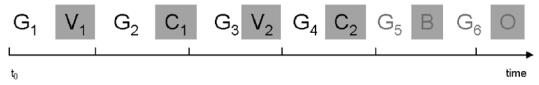

Fig. 10. Four sections: $\text{Verse}_1(V_1), \text{Chorus}_1(C_1), \text{Verse}_2(V_2), \text{Chorus}_2(C_2)$ and four gaps: G_1, G_2, G_3, G_4 in a popular song. G_1 is the Intro before start of V_1. G_2, G_3, G_4 are the gaps between the sections. Bridges and Outros are not currently aligned by LyricAlly.

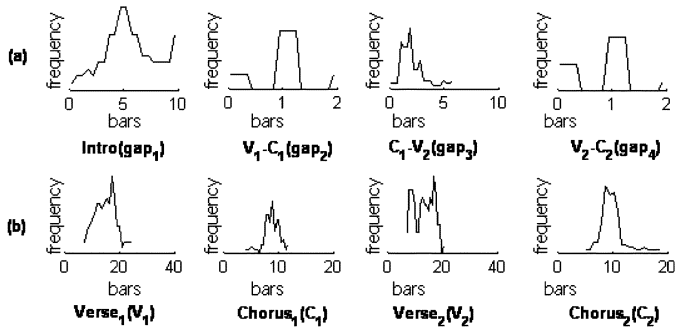

Fig. 11. Duration distributions of (a) nonvocal gaps, (b) different sections of the popular songs with $V_1 - C_1 - V_2 - C_2 - B - O$ structure. X-axis represents duration in bars. Note that gap durations have a smaller range compared to section durations.

Our algorithm is heuristically driven and logically consists of two steps:

- section-level alignment, which uses the chorus and vocal detectors outputs to demarcate the section boundaries;
- line-level alignment, which uses the vocal detector, line duration estimation, and calculated section boundaries as input to demarcate individual line boundaries.

A. Section-Level Alignment

We observe that the audio processing components have different levels of accuracy. As the repetition-based chorus detector is more accurate than the vocal detector, we use this component as an anchor to determine the remaining alignment. The chorus detector provides the start and end points for both choruses in the input. Section-level alignment is thus equivalent to finding the boundaries of the remaining two verse sections.

Recall that the input songs to LyricAlly must have a structure consisting of $\text{Verse}_1(V_1)$, $\text{Chorus}_1(C_1)$, $\text{Verse}_2(V_2)$, and $\text{Chorus}_2(C_2)$ sections. A dual representation of this structure focuses on the instrumental gaps, rather than the sung sections. We denote this representation with gaps G_1, G_2, G_3, and G_4, where gaps occur between the beginning of the song and the first verse (G_1) and between each of the four sections $(G_2\text{-}G_4)$, as illustrated in Fig. 10. From an analysis of the songs in our testbed, we observe that the section durations are highly variable, but that gap durations are more stable and have a smaller range (gap lengths are usually small multiples of the bar, whereas section lengths are usually much longer).

The start and end points for the verse sections are bounded by the choruses and the gap length distributions. For example, Verse_1 must start after the beginning of the song (t_0) but before $t_0 + 10$ bars (as the maximum gap length observed for G_1 is ten bars, as shown in line (a) of Fig. 11), and Verse_2 must start after the end of $\text{Chorus}_1(C_{1E})$ but before $C_{1E} + 5$ bars (the maximum gap length for G_3).

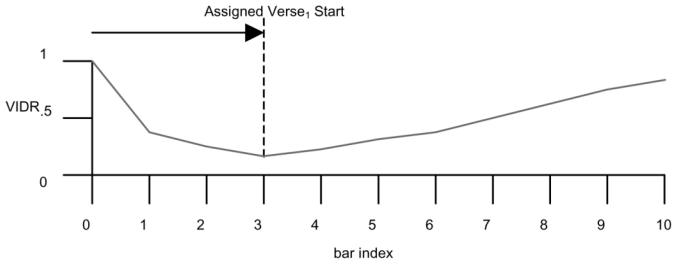

Fig. 12. Forward search in Gap_1 using VIDR to locate the start of Verse_1. Windows are grown by four IBIs (one bar) at each iteration.

Let us consider the specific process of detecting the start of Verse_1. We search forward within the window $[t_0, t_0 + 10 \text{ bars}]$ (the window which covers the gap G_1) starting from the first IBI for the most likely point where Verse_1 starts using the vocal detector. The intuition is that gaps contain only instrumental sound while Verse_1 contains vocal singing; we reduce the problem into finding a point where vocals are most likely to start. If the vocal detector had high accuracy, we could search for the first interbeat interval classified as vocal, and equate it with the start of the verse. However, the vocal detector is inaccurate and sometimes reports spurious results, necessitating postprocessing.

We define the *vocal to instrumental duration ratio* (VIDR) metric to assist with this postprocessing. It is defined as the ratio of vocal to instrument probability in a time window

$$\text{VIDR}(start, end) = \frac{\text{number of vocal IBIs in } span(start, end)}{\text{number of non-vocal IBIs in } span(start, end)}$$

where IBI is the interbeat interval for which we classify as vocal or nonvocal. We calculate the VIDR for a series of time windows, anchored at the start point of the search and growing towards the end point. Specifically, we start with a length of one bar (four IBIs), and over ten iterations, grow the window by one bar each time, in the direction of the end point. The change in VIDR over a growing window of IBIs indicates the growing or diminishing presence of vocals. Normally, when the window is in a gap, the VIDR is low, close to 0. When the window gradually moves into the verse and IBIs are classified as vocals, the VIDR increases. We mark the window where the VIDR starts to increase as the verse starting point.

Fig. 12 plots VIDR as it changes over the ten windows, showing that the VIDR starts to increase after the fourth window. We make two observations: first, the beginning of the verse is characterized by the strong presence of vocals that causes a rise in the VIDR over subsequent windows; second, as the vocal detector may erroneously detect vocal segments within the gap (as in the first window), the beginning of the verse may also be marked by a decline in VIDR over previous windows. We thus use the global minimum in the VIDR curve as the V_1 start point. Note that we only calculate VIDR at bar boundaries as we assume vocals start and end at bar boundaries as well.

We conduct a similar forward search to find the start point for V_2 using G_3 and the end point of C_1. In a similar manner, backward searches are conducted to determine the end points

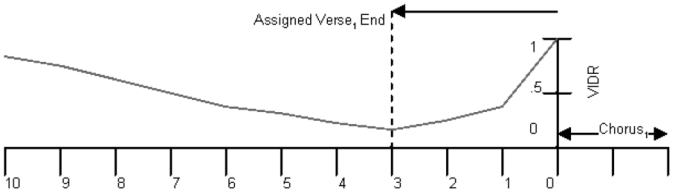

Fig. 13. Backward search to locate the ending of a verse.

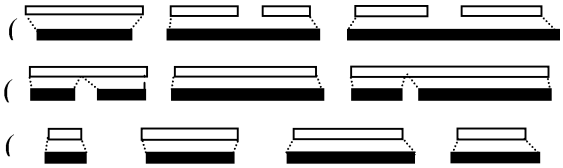

Fig. 14. (a) Grouping, (b) partitioning, and (c) forced alignment. White rectangles represent vocal segments (from vocal detector), and black rectangles represent durations of lyric lines (given by text processor).

of verse sections, as shown in Fig. 13. Once both starting and ending points of sections are determined, we calculate section lengths which are used in line level alignment, described next.

B. Line-Level Alignment

In line-level alignment, we combine the line duration estimates from text processing with vocal detection results. The text processor is fairly accurate in estimating line duration but is incapable of providing starting point of a line. The vocal detector is able to detect the presence of vocals in the audio—which can be used as evidence of line starting points—but does not accurately determine line durations. These complementary strengths are combined in line-level alignment.

The objective of line alignment is to localize each lyric line within the section start and end points given by section alignment. We take the number of lines given by the text lyrics as the target number of vocal segments to find in the audio. As the vocal detection output may not match the target number, we force align the vocal detection output to map one-to-one to the number of lyric lines when necessary. When the number of vocal segments is greater or less than the target, we need to perform grouping or partitioning of the vocal segments. This process is illustrated in Fig. 14.

First, we normalize both duration times from the vocal detection and the text processing section results to sum to 1. Next, we consider different ways to partition or group detected vocal segments to match the target number of lyric lines. We consider a partition or a grouping optimal when the resulting normalized vocal segment durations best match the normalized text durations. This is calculated by minimizing the absolute difference between each line's text duration and its corresponding vocal segment duration, while constraining the number of edits made (grouping or partitioning).

The system then combines both the text and vocal duration estimates to output a single, final estimate of line duration. We start by calculating the final section duration as

$$D_{\text{final}} = \max(D_{\text{text}}, D_{\text{vocal}}) - \alpha|D_{\text{text}} - D_{\text{vocal}}|.$$

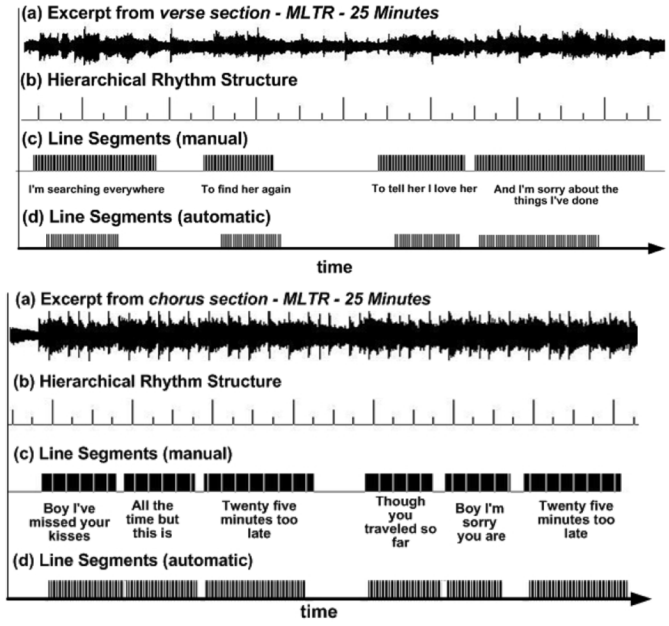

Fig. 15. Line segment alignment for (top) a verse section and (bottom) a chorus section.

Here, α is optimized empirically such that the total duration of the vocal segments within each section is closest to the section duration estimates, calculated in the previous section. The final duration of individual lines $d_{\text{final}}(i)$ are calculated as a ratio of this section length, specifically

$$d_{\text{final}}(i) = \left(\frac{d_{\text{text}}(i)}{D_{\text{text}}}\right) \times D_{\text{final}}, \quad i = 1, 2, \ldots, L$$

where L is total number of lines in the section. Note that only the text estimates for the line are used explicitly in the formula, as we have them to be more reliable. Example results of our automatic synchronization for sample lines in the chorus and verse are shown in Fig. 15.

VII. Evaluation

We have evaluated LyricAlly on our testbed of 20 popular songs, for which we manually annotated the songs with starting and ending timestamps of each lyric line. When possible, we observe training and testing separation by using leave-one-out cross-validation (i.e., train on 19 songs' data, test on 1, repeat 20 times, and average the results). Past work used random sampling to compute these alignment errors. In contrast, we evaluate our system over our entire testbed.

Evaluation in past audio alignment work [5] computes an alignment error using the mean and standard deviation of the error in estimating the start point and duration, given in seconds. We note that error given in seconds may not be ideal, as a one-second error may be perceptually different in songs with different tempos. We suggest measuring error in terms of bars as a more appropriate metric. We show alignment error measured in bars in Fig. 16 for both section and line-level alignments.

In analyzing these results, we see that starting point calculation is more difficult than duration estimation for individual lines. This is likely because the starting point is derived purely

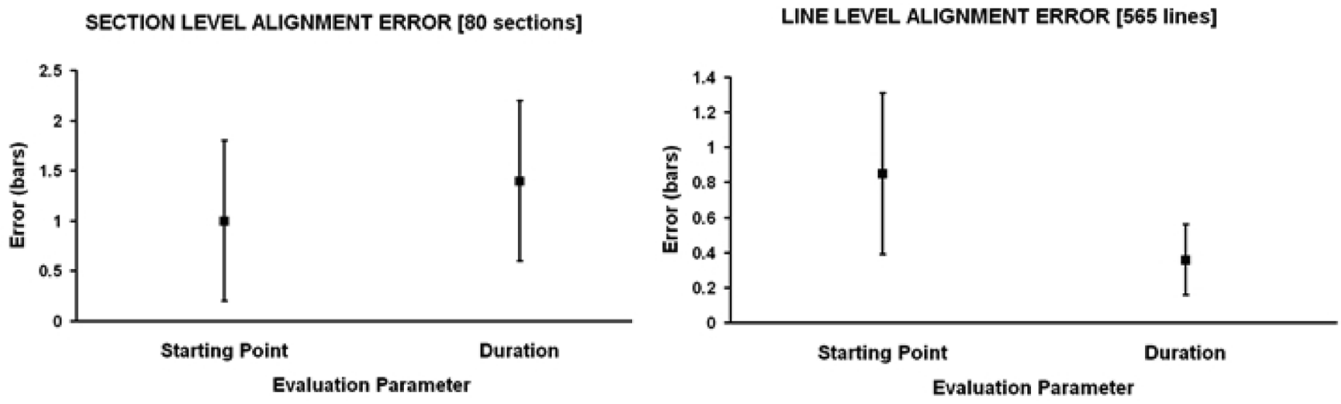

Fig. 16. Section- and line-level alignment error in bars.

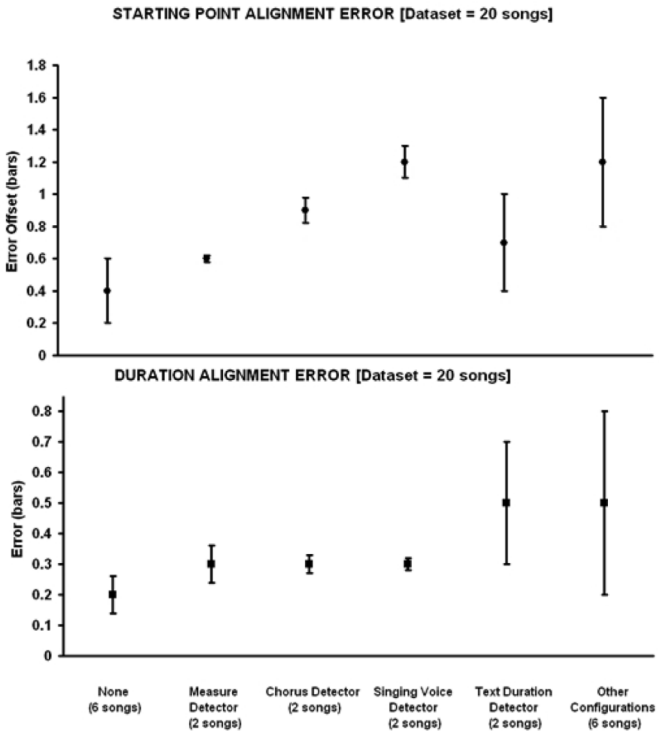

Fig. 17. Absolute alignment error over all lines ($n = 565$) in the dataset. The number of songs where the specific component fails is indicated in parentheses.

by audio processing, whereas the text processing assists in the duration estimation. We also see that durations of entire sections are more variable than single lines, as sections are larger units. On the other hand, the performance of starting point estimation does not vary significantly between lines and sections.

A. Error Analysis of Individual Components

As LyricAlly is a prototype based on an integration of separate components, we also want to identify critical points in the system. Which component is the bottleneck in system performance? Does a specific component contribute more error than others in start point localization or in estimating duration?

To answer these questions, we need to analyze each component's contribution to the system. We reanalyze our system's performance by categorizing cases in which one component fails. As expected, the system works best when all components perform well, but performance degrades gracefully when certain components fail.

Different components are responsible for different types of errors. If we classify starting point and duration calculations as either good or poor, then we have four possible scenarios for a song's alignment, as exemplified in Fig. 18.

Failure of the rhythm detector affects all components as estimates are rounded to the nearest bar, but the effect is limited to beat length over the base error (row 1 in Fig. 17). The text processor is only used to calculate durations, and its failure leads to less accurate line level duration, as in line (b) of Fig. 18. Failure of the chorus detection causes the starting point anchor of chorus sections to be lost, resulting in cases such as line (c) of Fig. 18. When the vocal detector fails, both starting point and duration mismatches can occur, as shown in line (d) of Fig. 18.

VIII. Discussion

These results indicate that each component contributes a performance gain in the overall system. Excluding any component degrades performance. If we weight starting point and duration errors equally, and equate minimizing the sum of squares of the per-line error as the performance measure, we can rank the components in decreasing order of criticality

$$\text{Vocal} > \text{Measure} > \text{Chorus} > \text{Text}.$$

We believe that errors in starting point and duration are likely to be perceived differently. In specific, starting point errors are more likely to cause difficulties for karaoke applications in comparison to duration errors. When we weight start point estimation five times more important than duration estimation, a different ranking emerges

$$\text{Chorus} > \text{Vocal} > \text{Text} > \text{Measure}.$$

We believe that this is a more realistic ranking of the importance of each of the components. As the components contribute differently to the calculation of start point and duration calculation, their effect on the overall system is different. As can be seen by our integration strategy, the accurate detection and alignment of choruses is paramount as it provides an anchor for subsequent verse alignment.

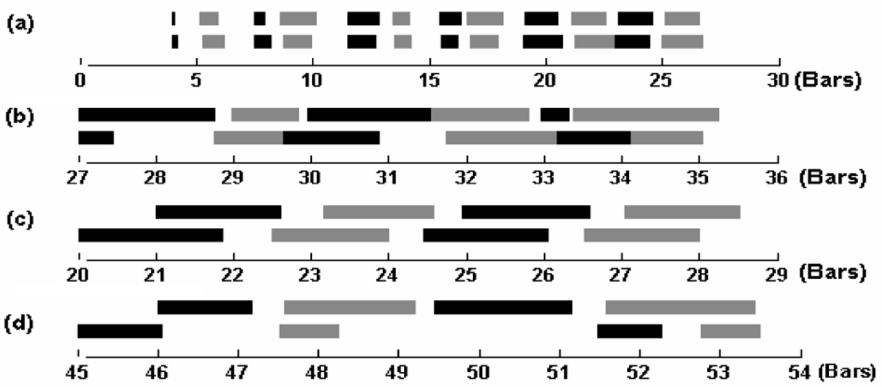

Fig. 18. Alignment between manual (upper line) and automatic timings (lower line). (a) Perfect alignment. (b) Duration mismatch. (c) Starting point mismatch. (d) Both duration and starting point mismatches.

While we have emphasized error analysis in our evaluation, it is not the only criteria in assessing performance. Efficiency is also paramount, especially for applications that may be deployed in mobile devices or in real-time applications. As the text processing of the dataset requires magnitudes less computation to perform as compared to the audio components, current work on LyricAlly hopes to push the use of such efficiently obtainable information earlier on in the architecture to achieve faster alignment processing.

It should be noted that we have adopted strong model assumptions to simplify implementation. We believe that the proposed approach can generalize to other scenarios, perhaps by determining the song structure first by preprocessing the audio and lyrics. These model assumptions are further enhanced by the use of heuristics based on music knowledge. We adopt this to correct the unsatisfactory performance of singing voice separation. In the future, when singing voice separation technology matures, lyric/signal alignment may be solved without the need for rhythm and chorus information.

IX. Conclusion and Future Work

Creating a usable music library requires addressing the description, representation, organization, and use of music information [17]. A single song can be manifested in a range of symbolic (e.g., score, MIDI, and lyrics) and audio formats (e.g., MP3). Currently, audio and symbolic data formats for a single song exist as separate files, typically without cross-references to each other [17]. An alignment of these symbolic and audio representations is definitely meaningful but is usually done in a manual, time-consuming process. We have pursued the alternative of automatic alignment for audio data and text lyrics, in the hopes of providing karaoke-type services with popular music recording.

To address this goal, we have developed LyricAlly, a multimodal approach to automate the alignment of text lyrics with acoustic musical signals. It incorporates state-of-the-art components for music understanding for rhythm, chorus, and singing voice detection. We leverage text processing to add constraints to the audio processing, pruning unnecessary computation, and creating rough estimates for duration, which are refined by the audio processing. LyricAlly is a case study that demonstrates that two modalities are better than one, and furthermore, demonstrates the need for processing of acoustic signals on multiple levels.

Our project has lead to several innovations in combined audio and text processing. In the course of developing LyricAlly, we have demonstrated a new chord detection algorithm and applied it to hierarchical rhythm detection. We capitalize on the rhythm structure preprocessing to improve the accuracy and efficiency of all components in LyricAlly. This is most effectively demonstrated in the chorus detection where a computational efficiency of 98% is gained over the previous state-of-the-art while maintaining similar accuracy. We have also developed a singing voice detection algorithm which combines multiple HMM models with bootstrapping to achieve higher accuracy. In our text processing models, we use a phoneme duration model based on singing voice to predict the duration of sections and lines of lyrics.

To integrate the system, we have viewed the problem as a two-stage forced alignment problem. A key observation is that each component gives different evidence for alignment and at different accuracy levels. In LyricAlly's architecture, we capitalize on this as much as possible. For example, the repetition-based chorus detection is quite accurate and serves to anchor the alignment process for verses. As such, we can limit the use of the more computationally intensive vocal detection to places where it is necessary; in particular, in a neighborhood of start time candidates of verses. Similarly, text processing yields a more accurate description and duration estimation of lyric lines—a constraint that is used to force align and filter the detected vocal segments.

LyricAlly is currently limited to songs of a limited structure and meter. Our hierarchical rhythm detector is limited to the common 4/4 time signature. Integration currently only aligns choruses and verses of songs. Furthermore, the chorus detector is constrained to produce exactly two chorus detections as this is the input specification to LyricAlly. In vocal detection, we could consider an implementation using mixture modeling or classifiers such as neural networks. These are important areas in the audio processing component for future work. These limitations are used to constrain the audio processing and achieve more accurate results: for example, knowing that there are two choruses in a song instead of three helps the chorus detector

prune inconsistent hypotheses. In this paper, we have presented LyricAlly as a holistic system. However, its components differ in terms of their individual limitations. For example, the text processing component can process input lyrics of arbitrary structure (i.e., songs with more or less chorus/verses, with intros, bridges and outros). Similarly, the chorus detector can be constrained to find additional choruses or outros, where the constraints may be provided by text processing. As LyricAlly is scaled up to handle more complex song structures, we feel that such synergies between text and audio processing will play a larger role.

REFERENCES

[1] V Arifi, M Clausen, F. Kurth, and M Muller, "Automatic synchronization of music data in score-, MIDI- and PCM-format," in *Proc. Int. Conf. Music Inf. Retrieval (ISMIR)*, 2003, pp. 219–220.

[2] A. Berenzweig and D. P. W. Ellis, "Locating singing voice segments within music signals," in *Proc. Workshop Appl. Signal Process. Audio Acoust. (WASPAA)*, 2001, pp. 119–122.

[3] A. Berenzweig, D. P. W. Ellis, and S. Lawrence, "Using voice segments to improve artist classification of music," in *Proc. AES-22 Int. Conf. Virt., Synth., Ent. Audio.*, Espoo, Finland, 2002, pp. 79–86.

[4] B. Chen, H.-M. Wang, and L.-S. Lee, "Improved spoken document retrieval by exploring acoustic and linguistic cues," in *Proc. Eurospeech*, 2001, pp. 1–4, CD-ROM.

[5] R. Dannenberg and N. Hu, "Polyphonic audio matching for score following and intelligent audio editors," in *Proc. Int. Comput. Music Conf. (ICMC)*, Singapore, 2003, pp. 27–33.

[6] "EU Project Semantic HIFI." [Online]. Available: http://shf.ircam.fr/

[7] M. Furini and L. Alboresi, "Audio-text synchronization inside MP3 files: A new approach and its implementation," in *Proc. IEEE Consumer Commun. Netw. Conf.*, Las Vegas, NV, 2004, pp. 522–526.

[8] M. Goto, "An audio-based real-time beat tracking system for music with or without drum-sound," *J. New Music Res.*, vol. 30, no. 2, pp. 159–171, Jun. 2001.

[9] M. A. Goto, "Chorus-section detection method for musical audio signals," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2003, pp. 1783–1794.

[10] D. Iskandar, W. Wang, M. -Y. Kan, and H. Li, "Syllabic level automatic synchronization of music signals and text lyrics," in *Proc. ACM Multimedia (MM)*, Santa Barbara, CA, Oct. 2006, pp. 659–662.

[11] P. Knees, M. Schedl, and G. Widmer, "Multiple lyrics alignment: Automatic retrieval of song lyrics," in *Proc. Int. Conf. Music Inf. Retrieval (ISMIR)*, 2005, pp. 564–569.

[12] F. Kurth, M. Mueller, A. Ribbrock, T. Roeder, D. Damm, and C. Fremerey, "A prototypical service for real-time access to local context-based music information," in *Proc. Int. Conf. Music Inf. Retrieval (ISMIR)*, 2004, pp. 34–37.

[13] C. H. Lee, F. K. Soong, and B. H. Juang, "A segment model based approach to speech recognition," in *IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1988, vol. 1, pp. 501–541.

[14] B. Logan, A. Kositsky, and P. Moreno, "Semantic analysis of song lyrics," in *Proc. IEEE Int. Conf. Mutimedia Expo (ICME)*, 2004, pp. 827–830.

[15] A. Loscos, P. Cano, and J. Bonada, "Low-delay singing voice alignment to text," in *Proc. Int. Comput. Music Conf.*, Beijing, China, 1999, pp. 437–440.

[16] J. P. G. Mahedero, A. Martinez, and P. Cano, "Natural language processing of lyrics," in *Proc. ACM Multimedia*, 2005, pp. 475–478.

[17] N. Minibayeva and J.-W. Dunn, "A digital library data model for music," in *Proc. ACM/IEEE-CS Joint Conf. Digital Libraries (JCDL)*, 2002, pp. 154–155.

[18] P. J. Moreno, C. Joerg, J.-M. Van Thong, and O. Glickman, "A recursive algorithm for the forced alignment of very long audio segments," in *Proc. Int. Conf. Spoken Lang. Process.*, 1998.

[19] T. L. Nwe, F. S. Wei, and L. C. De Silva, "Stress classification using subband based features," *IEICE Trans. Inf. Syst.*, vol. E86-D, no. 3, pp. 565–573, 2003.

[20] N. Orio and D. Scwarz, "Alignment of monophonic and polyphonic music to a score," in *Proc. Int. Comput. Music Conf.*, San Francisco, CA, 2001, pp. 155–158.

[21] A. Shenoy, R. Mohapatra, and Y. Wang, "Key determination of acoustic musical signals," in *Proc. Int. Conf. Multimedia Expo (ICME)*, Taipei, Taiwan, 2004, CD-ROM.

[22] R. J. Turetsky and D. P. W. Ellis, "Ground truth transcriptions of real music from force-aligned MIDI syntheses," in *Proc. Int. Conf. Music Inf. Retrieval (ISMIR)*, 2003, pp. 135–141.

[23] G. Tzanetakis, "Song-specific bootstrapping of singing voice structure," in *Proc. Int. Conf. Multimedia Expo (ICME)*, Taipei, Taiwan, 2004, CD-ROM.

[24] J.-M. Van Thong, P. J. Moreno, B. Logan, B. Fidler, K. Maffey, and M. Moores, "Speechbot: An experimental speech-based search engine for multimedia content on the Web," *IEEE Trans. Multimedia*, vol. 4, no. 1, pp. 88–96, Mar. 2002.

[25] H. D. Wactlar, A. G. Hauptmann, and M. J. Witbrock, "Informedia: News-on-demand experiments in speech recognition," in *Proc. DARPA Speech Recognition Workshop*, 1996.

[26] C. K. Wang, R. Y. Lyu, and Y. C. Chiang, "An automatic singing transcription system with multilingual singing lyric recognizer and robust melody tracker," in *Proc. Eurospeech*, Geneva, Switzerland, 2003, pp. 1197–1200.

[27] Y. Wang, M.-Y. Kan, T. L. Nwe, A. Shenoy, and J. Yin, "LyricAlly: Automatic synchronization of acoustic musical signals and textual lyrics," in *Proc. ACM Conf. Multimedia*, 2004, pp. 212–219.

[28] Y. Wang, J. Tang, A. Ahmaniemi, and M. Vaalgamaa, "Parametric vector quantization for coding percussive sounds in music," in *Proc. Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2003, pp. 652–655.

[29] I. Waugh, "Song structure," *Music Tech Mag.*, vol. 7, pp. 62–63, Oct. 2003 [Online]. Available: http://www.musictechmag.co.uk, Ten Minute Master No. 18: Song Stucture.

[30] R. Weide, "CMU Pronouncing Dictionary (Rel. 0.6, 1995)." [Online]. Available: http://www.speech.cs.cmu.edu/cgi-bin/cmudict)

[31] H. Yang, L. Chaisorn, Y. Zhao, S.-Y. Neo, and T.-S. Chua, "VideoQA: Question answering on news video," in *Proc. ACM Multimedia*, 2003, pp. 632–641.

[32] Y. Zhu, K. Chen, and Q. Sun, "Multimodal content-based structure analysis of karaoke music," in *Proc. ACM Multimedia*, 2005, pp. 638–647.

Min-Yen Kan is an Assistant Professor at the National University of Singapore (NUS). His research interests include document structure acquisition, verb analysis, scholarly digital libraries, and applied text summarization. Before joining NUS, he was a Graduate Research Assistant at Columbia University, New York, and had interned at various industry laboratories, including AT&T, IBM, and Eurospider Technologies, Switzerland.

Ye Wang received the B.Sc. degree in wireless communications from the Southern China University of Technology, Guangzhou, China, in 1983, the Diplom-Ingenieur (M.Sc.) degree in telecommunications from Technische Universität Braunschweig, Braunschweig, Germany, in 1993, and the Lic.-Tech. and Dr.-Tech. degrees in information technology from the Tampere University of Technology, Tampere, Finland, in 2000 and 2002, respectively, during a full-time employment at Nokia Research Center as a Research Staff Member.

In 2001, he spent a research term at the University of Cambridge, Cambridge, U.K. Since 2002, he has worked as an Assistant Professor at the Department of Computer Science, National University of Singapore. His research interests include audio processing, error resilient audio content delivery over wired/wireless packet networks, low-power media processing for handheld devices, and multimedia-based music processing. He holds several patents in these areas and has published more than 30 refereed international journal and conference papers.

Denny Iskandar is currently pursuing the M.Sc. degree from the National University of Singapore.

He was a Research Officer in the Speech and Dialogue Processing Group at the Institute of Infocomm Research, Singapore. He is currently with G Element, Singapore.

Tin Lay Nwe received the B.E. degree in electronics from the Yangon Institute of Technology, Yangon, Myanmar, in 1994 and the Ph.D. degree in electrical engineering from the National University of Singapore in 2004.

She was a Senior Engineer from 1994 to 1998 in Myanmar Daewoo Electronics Co., Ltd., Yangon. In October 2003, she joined the School of Computing, National University of Singapore, as a Research Assistant. In June 2004, she joined the Institute for lnfocomm Research, Singapore, where she is now a Research Fellow. Her current research interests include robust speech recognition, rich transcription, and music information retrieval.

Arun Shenoy received the M.S. degree from the National University of Singapore in 2004.

Prior to graduation, he joined the National University of Singapore as a Research Assistant working on audio processing. He has published several papers in both IEEE and ACM venues on the subject of key detection and singing voice detection in audio.