Content-Based Tools for Editing Audio Stories

Steve Rubin*, Floraine Berthouzoz*, Gautham J. Mysore†, Wilmot Li†, Maneesh Agrawala*

*University of California, Berkeley
{srubin,floraine,maneesh}@cs.berkeley.edu

†Adobe Research
{gmysore,wilmotli}@adobe.com

ABSTRACT

Audio stories are an engaging form of communication that combine speech and music into compelling narratives. Existing audio editing tools force story producers to manipulate speech and music tracks via tedious, low-level waveform editing. In contrast, we present a set of tools that analyze the audio content of the speech and music and thereby allow producers to work at much higher level. Our tools address several challenges in creating audio stories, including (1) navigating and editing speech, (2) selecting appropriate music for the score, and (3) editing the music to complement the speech. Key features include a transcript-based speech editing tool that automatically propagates edits in the transcript text to the corresponding speech track; a music browser that supports searching based on emotion, tempo, key, or timbral similarity to other songs; and music retargeting tools that make it easy to combine sections of music with the speech. We have used our tools to create audio stories from a variety of raw speech sources, including scripted narratives, interviews and political speeches. Informal feedback from first-time users suggests that our tools are easy to learn and greatly facilitate the process of editing raw footage into a final story.

Author Keywords

Audio editing, storytelling, transcript-based editing, music browsing, music retargeting

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation (e.g. HCI): User Interfaces - Graphical user interfaces

INTRODUCTION

Audio stories are an engaging form of communication that are commonly heard in podcasts, radio programs and audiobooks. But, creating a compelling audio story requires careful editing and production. Starting from the raw recorded speech, experienced producers select and combine the most salient content and then refine individual sentences to improve the phrasing and rhythms of the speech. Many audio stories also include a musical score that plays under the speech. Producers choose music that enhances the emotion of the story and then adjust the length and volume of musical sections to both complement and emphasize key moments in the speech. In short, creating an effective audio story requires making a number of high-level editing and design decisions that together define the narrative arc and emotional tone of the story [7, 8, 20, 24, 33].

Most existing audio editing systems provide all of the functionality necessary to support such editing and production tasks. However, these systems force producers to manipulate the speech and music at the level of the audio waveform. As a result producers must map their high-level story editing and design goals onto a sequence of low-level editing operations — e.g. selecting, trimming, cutting and moving sections of a waveform. Manually applying each of these low-level edits is often tedious and usually very time-consuming.

In this paper we present a set of tools that analyze the audio content of raw speech and music tracks and thereby allow producers to work at a much higher level. Our tools address challenges that span the process of creating audio stories, from (1) navigating and editing speech, to (2) selecting appropriate music for the score, and (3) editing the music to complement the speech. Using these tools the producer can focus on developing the content and emotion of the story while our system automatically applies the appropriate low-level operations on the audio waveforms.

Our audio editing system includes the following key features:

Transcript-based speech editing. To help producers navigate and edit raw speech recordings, our interface includes a transcript view of each speech track. As in previous transcript-based speech editing systems [10, 14, 40], producers can directly modify the transcript text, and our system propagates the corresponding edits to the speech waveform. However, unlike previous systems, our transcript view groups similar sentences, so that the producer can quickly listen to different versions of a line and insert the best one into the story. The transcript view also marks pauses and breaths in the speech to help the producer refine phrasings while maintaining a natural sounding rhythm.

Multi-feature music browsing. To help producers find the appropriate music to score a story, our system includes a music browser that supports searching for songs based on several different features, including emotional characteristics, tempo, key, or a crowdsourced set of keywords associated with the music. In some cases, a producer may need several different songs that sound similar to form a coherent score that does not sound repetitive. For these situations, our browser provides a similarity search that takes a single song as input and finds other songs that are similar in timbre.

Structure-based music editing. To help producers integrate music into a story, our system includes two types of music

UIST '13, October 8–11, 2013, St. Andrews, United Kingdom.
ACM 978-1-4503-2268-3/13/10.
http://dx.doi.org/10.1145/2501988.2501993

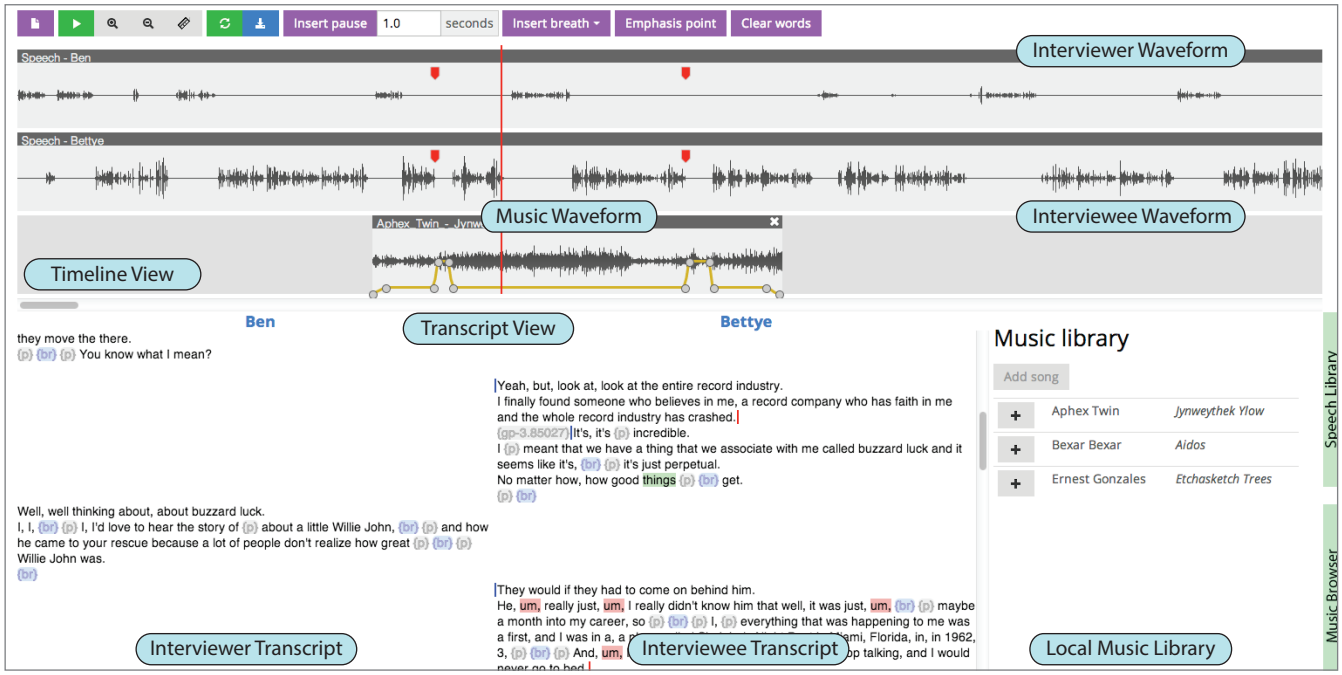

Figure 1. Our editing interface features two views of each speech track: a traditional waveform view, and a text-based transcript view.

editing tools that make it easy to retarget music to fit the speech. Our *simple music retargeting tool* allows the producer to interactively extend or shorten a section of music while maintaining a natural sounding result. Our *constrained retargeting tool* automatically restructures a song to match a desired length while also highlighting one or more emphasis points in the speech.

We have used our tools to create audio stories from a variety of speech sources, including scripted narratives, interviews, and presidential speeches. We also report on informal evaluation where we asked users to create audio stories from raw speech recordings using our system. The results demonstrate that our tools enable first-time users to produce high-quality stories that contain a variety of edits to raw speech and music. User feedback suggest that our tools are easy to learn and greatly facilitate the production of audio stories.

RELATED WORK

Traditional digital audio workstations (DAWs) like Adobe Audition [1] and Avid ProTools [2] allow producers to edit waveforms in a multi-track timeline interface. These systems provide a wide variety of low-level signal processing tools [11, 13, 28, 44] for manipulating audio waveforms, but their interfaces are very complex. Moreover, because they are designed to serve as general-purpose audio production systems, they include many features that are not directly relevant for creating audio stories.

Hindenburg Systems [5] develops tools that are designed specifically for producing audio stories, stripping away much of the complexity of full-fledged DAWs while simplifying common tasks in audio journalism like managing the relative volume levels of the speech and music tracks. Despite these usability improvements, Hindenburg still adheres to the standard waveform-based metaphor of audio editing and forces producers to directly edit the waveform.

Researchers have investigated several alternatives to direct waveform-based navigation and editing. Barthet et al. [9] segment podcasts into speech and music so that listeners can jump to the different sections. Fazekas et al. [17] split songs into verse and chorus to similarly enable quick navigation. In the context of video editing, researchers have developed tools that leverage higher-level structural annotations. Davis [15] proposes video editing tools that make extensive use of metadata (e.g. camera settings and semantic annotations) to describe the content of the raw footage. Users can then browse and rearrange the footage through an iconic representation of the metadata. Li et al. [25] enable faster browsing of video clips by automatically marking shot boundaries and enabling pitch-preserving rapid audio playback. Likewise, Girgensohn et al. [19] present a system that automatically filters out unsuitable video clips based on an analysis of the camera motion. Unlike these systems, our work focuses on using a transcript of the speech track to navigate and edit audio stories.

We build on several previous techniques for text-based navigation and editing. Whittaker and Amento [40] show that users strongly prefer editing voicemail through a transcript instead of a waveform, even when automatic speech recognition produces significant errors in the transcript. However, because they focus on voicemail recordings, their work did not investigate how to produce high-quality edited output. Casares et al. [14] and more recently, Berthouzoz et al. [10] present video editing systems that include speech-aligned transcript editors to enable more efficient navigation and editing of video. We extend this approach to the do-

main of audio editing and provide a number of additional transcript-based editing tools (e.g. grouping similar sentences and editing breaths).

Researchers have also developed tools for browsing large collections of music based on a variety of low-level features (i.e., tempo, timbre, volume, etc.) [26, 31, 30]. However, these tools are designed to serve as general-purpose music browsers for music discovery and designing playlists. None of them integrate into an audio editing system.

Example-based music synthesis and retargeting is an active area of research [22]. Lu et al. [29] describe a method for generating *audio textures* that takes an audio track as input and reshuffles its frames to generate a similar-sounding track. Zils and Pachet's [43] Musical Mosaicing system synthesizes a target waveform by sequencing snippets from a database of music. The EchoNest, a company that provides a robust API for music analysis, has demonstrated tools for extracting the structure of a song and then retargeting it to play indefinitely [23, 39]. In concurrent work, Wenner et al. [38] have developed a music retargeting algorithm that preserves user-specified constraints. Our music retargeting tools similarly consider the structure of a music track to extend or shorten it in a natural sounding manner. However, unlike previous techniques we also impose retargeting constraints based on combined features of the music and speech tracks.

AUDIO EDITING INTERFACE

We have developed a set of tools that are designed to help producers edit raw speech and music tracks into audio stories. We first describe the interface to these tools (Figure 1) and then present the content-based analysis algorithms that enable each tool in the ALGORITHMIC METHODS section.

Transcript-based speech editing

The producer starts by loading raw speech tracks into the editing interface. In Figure 1, the producer is editing the two tracks of an interview between Ben Manilla and recording artist Bettye Lavette. Our system displays two different views of each track: the timeline view shows the audio waveform, and the transcript view shows the corresponding text. Each speaker's transcript appears in a separate column of the transcript view and the text alternates between the columns as the speakers talk back and forth.

Each transcript is time-aligned with the corresponding waveform, so that selections and edits made to the text are reflected in the waveform and vice versa. All edit operations (i.e. cut, copy, paste, delete) in the transcript view occur at the word level (Figure 2). Our system snaps selections to the nearest word boundary to prevent the producer from breaking words into fragments. The transcript view helps the producer quickly navigate to specific parts of a speech track and edit the content of the story.

With interviews it is essential for the speech tracks to stay in sync with one another as the producer makes edits. Our system provides a linked editing mode that automatically maintains such synchronization. If the producer deletes part of a track for one speaker, our system deletes the corresponding

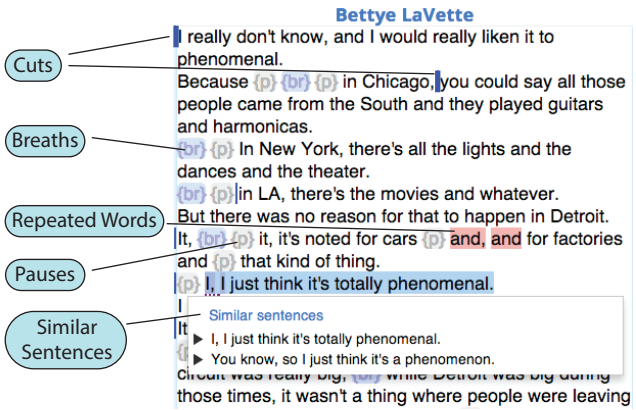

Figure 2. The transcript view allows producers to edit the story at the word level. This view marks cuts, breaths, pauses, repeated and unnecessary words, and similar sentences, all of which enable the producer to quickly edit the speech.

region from the waveform and transcript of all linked tracks. Similarly if the producer inserts text into the transcript of one speaker, our system inserts background *room tone* into the corresponding regions for the linked speakers. Producers usually capture room tone at the start of a recording session and our system automatically treats relatively silent segments from the beginning of the track as room tone.

Speakers often record multiple versions (or takes) of the same sentence to try variations in voicing or wording. Producers must then choose the most appropriate take for the story. Our system analyzes the transcript to identify retakes and underlines sentences in the transcript view for which similar alternatives are available. Clicking on the underline opens a drop-down showing the similar alternatives (Figure 2). The producer can listen to any of these takes and select her favorite without having to search for retakes through the entire raw recording.

After editing together a rough cut of the raw tracks, the producer next focuses on refining individual sentences to improve the flow of the speech. Our system identifies 'uhs', 'ums' and repeated words and highlights them in red so that the producer can easily delete them (Figure 2).

The transcript view also explicitly marks the breaths and pauses that occur in each speech track (Figure 2). These tokens help the producer maintain the natural patterns of speech as she edits the story. For example, speakers typically take a breath and pause for a moment before uttering each new sentence [7]. After rearranging sentences in the transcript view the producer can immediately check that the breath-pause combination occurs between sentences. The producer can also split a long sentence into two sentences by typing a ('.') in the transcript view, and our system automatically inserts the breath-pause combination before the next sentence. Finally, the producer can manually add or remove breaths and pauses as necessary. Pauses often serve to emphasize the preceding speech and give the listener time to reflect on what was said. Thus, the producer may choose to insert pauses at different points in the speech, to emphasize particular thoughts

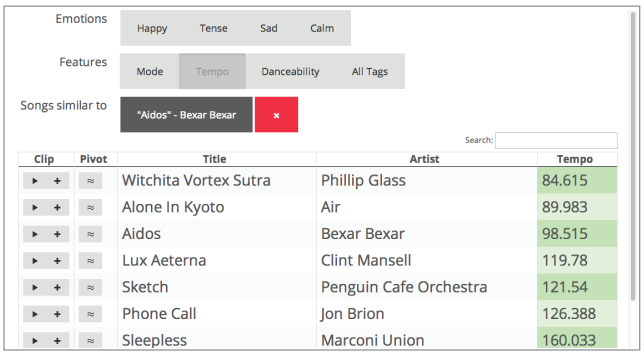

Figure 3. The music browser allows producers to filter by emotion, find similar sounding songs, and sort by low-level feature.

that the speaker may have originally rushed through. The default pause length is 250 ms, and the producer can manually adjust its length as necessary. Our system fills the pause with background room tone.

Multi-feature music browsing

Producers often add a musical score to an audio story in order to emphasize specific moments or emotions [7, 8]. To help producers find the appropriate music, our system provides a music browser (Figure 3). By default, our browser includes a collection of 90 songs that are commonly used in radio stories, but the producer can add other songs to the browser as well. Each song listing includes a '▶' button, to play a snippet of the song. When the producer finds a song that she likes, she uses the '✚' button to add the song to the local music library, which appears to the right of the transcript view (Figure 1). If the producer has a specific song in mind, she simply searches for the song by name. Otherwise, the producer can search for music in several different ways.

Finding music that fits an emotion

Producers often use a musical score that matches the emotion of a particular moment in the speech. Our music browser allows the producer to quickly find emotionally appropriate music by coarsely filtering the songs based on a measure of their emotion. Specifically, we use Russell's circumplex model of 'valence' and 'arousal' to measure the emotion of the songs [36]. Valence ranges from "misery" to "pleasure," and arousal ranges from "sleepy" to "activated." With our browser, the producer can filter the music to only see songs that corresponds to one of the four quadrants of the valence/arousal plane (Figure 4). For example, to find happy and upbeat music, the producer can select the high valence, high arousal quadrant Q1.

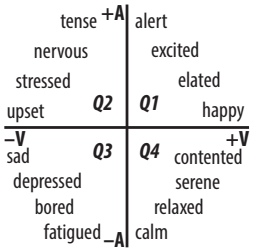

Figure 4. Russell's circumplex model of emotion.

Finding similar-sounding music

In some cases, the producer may wish to find a few different songs that all sound similar in order to form a musical score that is coherent but does not repeat songs. Using our browser, the producer first finds one song that she likes and then clicks the '≈' button. The browser then sorts the library to show the songs that are closest in timbre to the selected song. Timbre reflects the instrumentation of a song and we have found it to be a good indicator sound similarity.

Searching by feature

Our browser also lets producers sort the music based on several low-level features that we compute from the music: *tempo*, *mode* (major or minor key), and *danceability* (a feature from The EchoNest [4]). For example, if the producer is looking for an up-tempo song for her story, she can sort in descending order based on tempo and listen to songs starting from the top of the list.

Searching by keyword

Producers can also search for music using descriptive keywords (e.g., find *chill* music). To support this type of search, our browser retrieves tags that have been assigned to songs by users of last.fm [6], a music-tracking social network with ∼50 million registered users and information on ∼45 million songs. Thus, the producer can simply type *chill* into a search box to find all the music that has been tagged with this keyword.

Structure-based music editing

Once the producer has added songs to the local music library, the next step is to combine the music with the speech track. We provide standard waveform-based editing tools that allow the producer to add a song to the timeline, trim and reposition it as necessary, and specify a spline to control the volume. However, we also provide a set of higher level tools that help the producer retarget music (i.e. extend or shorten it) to match the length of a corresponding speech segment.

Simple music retargeting

At times producers may find a segment of music that is appropriate for a particular section of the speech, but is the wrong length. If the music is too short, the producer first select the subsegment she would like to extend and then clicks the '↻' button that appears on the waveform. Our system automatically extends the music by adding a seamless loop to the selected subsegment. The producer can click this button as many times as necessary to bring the music to the desired length (Figure 5). In order to minimize repetition in the music, our system finds and adds the longest loop that occurs

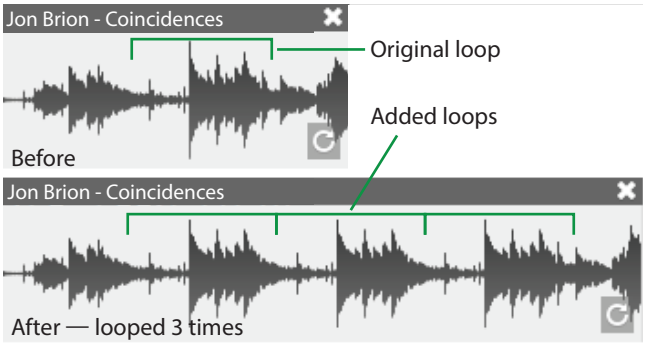

Figure 5. A musical segment before and after simple retargeting. Our tool finds a loop in the orginal segment (before) and in this case the producer repeats the loop three times in the final segment (after).

within the selected subsegment. Likewise, if the music is too long, the producer can click the '↻' button to remove audio from the selected subsegment.

Constrained music retargeting

Music usually contains a small set of *change points* that mark significant transitions in timbre, harmonic structure, or volume. Producers often emphasize a moment in the story by adding a *musical underlay* that aligns a music change point with an important point in the speech. Such underlays fade in the music before the emphasis point in the speech, then pauses the speech while the music solo plays at full volume, and finally fades out the music as the speech resumes. Rubin et al. [34] recently showed how to automatically detect change points in music and presented a semi-automated tool for creating a musical underlay that aligns one change point with one speech emphasis point. Our system includes this functionality – the producer marks an emphasis point in the speech, selects a song, and the system automatically produces the underlay.

In some cases, however, the producer may wish to use a single piece of music to continuously score a section of speech containing multiple speech emphasis points (Figure 6). In this situation the producer must align each emphasis point with a different music change point without stopping the music. Yet, the time between music change points rarely matches the time between speech emphasis points. Given multiple emphasis points and a song, our system automatically retargets the music by analyzing its structure and generating a new version that meets the alignment constraints.

ALGORITHMIC METHODS

Our audio editing tools all rely on algorithms that analyze the content of the raw speech and music tracks.

Obtaining the transcript

We obtain a transcript of the raw speech tracks using the crowdsourced transcription service, CastingWords.com [3]. It costs $1.00–$2.50 per minute of audio, depending on the desired turnaround time. By default the service produces *sanitized* transcripts that excludes words like 'uh', 'um', 'so' etc. However, our audio editor is designed to highlight such words in the transcript and let producers decide whether or not to remove them. Therefore we specify that the transcription service should produce *verbatim* transcripts. While automatic speech recognition software is continually improving, we have found that they cannot match the quality of crowdsourced transcription.

Aligning the transcript to the speech track

Once we have obtained the verbatim transcript of the speech track, we align it to the audio using the Penn Phonetics Lab Forced Aligner (P2FA) [42], which is built on the HTK (HMM toolkit) speech recognition software [41]. This aligner computes perceptual linear prediction (PLP) features to model each phoneme in the speech audio and applies a dictionary of word-to-phoneme translations on the text. It then uses the Viterbi algorithm for hidden Markov models to compute the maximum likelihood alignment between the speech and transcript. The aligner also inserts "pause" tokens into the transcript when there is a silent gap in the speech.

In practice we have found that transcripts frequently contain some words that are not in the P2FA word-to-phoneme dictionary (e.g. proper names, jargon, etc.). Therefore, we have extended the P2FA algorithm so that whenever it encounters such an unknown word, it uses the CMU Sphinx Knowledge Base Tool [35] to algorithmically determine the word's pronunciation. Although the resulting phonemes may be incorrect, we have found that the aligner is far more accurate when it has phonemes for every word in the transcript than when it is missing phonemes for some words.

Detecting breaths

Audible breaths are subtle but important elements in the natural rhythm of speech. The P2FA tool contains a model of breaths and can successfully align transcripts that contain explicit tokens indicating breaths. However, our crowdsourced transcripts do not indicate breaths because it is difficult for human transcribers to detect them reliably. Nevertheless we have developed an automated procedure that uses the initial transcript alignment and P2FA to detect breaths.

We assume that breaths can only occur in the segments already labeled as pauses in the initial alignment. Using P2FA we align each pause with a transcript containing a single breath token. The aligner returns a score indicating its confidence that the alignment is correct. If the score is greater than an empirically chosen threshold and the detected breath is longer than 100 ms then we accept the alignment and insert the breath token into the transcript.

If the producer manually adds a breath to the transcript we randomly select one of the detected breaths and insert the corresponding breath waveform into the timeline.

Detecting multiple takes of a sentence

Speakers sometimes record multiple takes of a sentence one after another. At other times they revisit lines later in a recording session. To identify such re-takes we cluster all of the sentences in the transcript so that each cluster contains all variations of a given sentence in the raw recording.

To build these clusters we first compute the similarity between each pair of sentences in the transcript using the Levenshtein string edit distance. If the edit distance is less than a threshold α we mark the pair as similar. Because the edit distance depends on the length of the strings, we have found that setting α to half the length of the longer sentence works well in practice. Intuitively, this threshold allows sentences marked as similar, to differ in about half of their characters. We then treat each sentence as a node in a graph and add an edge between each pair of similar sentences. Each connected component of this graph forms a cluster of similar sentences. When a producer clicks on an underlined sentence in the transcript view we show all of the sentences in the corresponding similarity cluster as the alternate takes.

Multi-feature music browsing

Our music browser contains a collection of songs and provides a set of tools designed to help producers quickly find

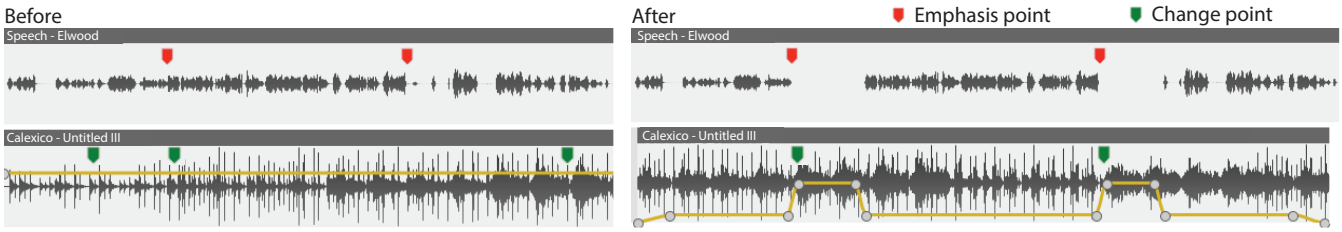

Figure 6. In constrained music retargeting, the producer selects multiple speech emphasis points (red markers) and a piece of music (before). Our system automatically identifies music change points (green markers) and aligns them with the emphasis points, by extending or shrinking the music as necessary, while preserving the local beat-level structure of the music (after).

appropriate music for scoring their audio stories. It relies on a set of five features (tempo, mode, danceability, timbre and valence/arousal) that characterize different aspects of each song in the collection.

We compute tempo and mode (major or minor key) using standard algorithms from music information retrieval and signal processing [16]. For the danceability feature we query the EchoNest [4] web service. They describe this feature as a combination of beat strength, tempo and tempo variability. We compute the timbre feature using mel-frequency cepstral coefficients (MFCCs) [27], which analyze the spectral characteristics of the song. For each song, we compute one 13-dimensional MFCC feature for a 15 second frame that begins 30 seconds into the song. Since MFCCs are a high dimensional feature we don't directly expose them to the producer. Instead when the producer searches for similar-sounding songs using the '≈' button we compute the nearest songs based on the Euclidean distance between their MFCCs. To support fast nearest-neighbor lookups, we store the MFCCs in a k-d tree.

The emotional controls in our browser filter songs based on the quadrants of the valence/arousal plane in Russell's circumplex model of affect [36] (Figure 4). Valence/arousal values are based on human perception, and can vary from person to person, so there is no direct way to compute them. Instead, following Schmidt et al. [37], we use a machine learning approach to estimate the valence/arousal for each song. The MoodSwings Turk dataset [37] contains a set of songs with corresponding valence/arousal values that were collected from workers on Amazon Mechanical Turk. We compute MFCCs for each of these songs and then perform multiple linear regression to build a mapping from the MFCCs to valence/arousal. Given a new song from our collection we first compute its MFCCs and then apply the regression model to predict valence/arousal values.

Structure-based music editing

Our interface allows the producer to extend or shorten a segment of music while preserving the local, beat-level structure of the song. To enable such music retargeting we first segment the song into beats using the approach of Ellis and Poliner [16]. We then consider how the beats are related to one another. A transition from beat m_i to beat m_{i+1} sounds natural because it respects the original progression of the music. To estimate whether a transition from beat m_i to beat m_j where $j \neq i+1$, would sound natural, we follow the approach of Jehan and Sundram [23] and compute a transition cost matrix T such that

$$T_{i,j} = \frac{D_h(m_{i+1}, m_j)}{\sigma_h} + \frac{D_t(m_{i+1}, m_j)}{\sigma_t}$$

where $D_h(m_i, m_j)$ is the Euclidean distance between the chroma features [16] (a measure of harmonics) for beats m_i and m_j and $D_t(m_i, m_j)$ is the distance between their MFCCs, while σ_h and σ_t are the standard deviations across all of the harmonic and timbre distances, respectively. Entries in this transition cost matrix are low for beats m_i and m_j if the timbre and harmonic features of beat m_j are close to the corresponding features of beat m_{i+1}. Thus, a low cost suggests a natural sounding transition.

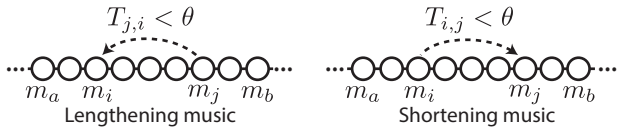

Figure 7. For simple music retargeting, we lengthen music by finding low cost transitions to earlier beats in the music. The producer can click a button to repeat such loops in the track. To shorten music, we find low cost transitions to later beats. The producer can than choose to delete the in-between beats.

We use this transition cost matrix T to enable simple and constrained music retargeting.

Simple music retargeting

In simple retargeting the producer can press a button to extend a segment of music by adding a loop to it. As shown in Figure 7, given a music segment from beat m_a to beat m_b we look for beats m_i and m_j such that $a \leq i < j \leq b$ and the transition cost from m_j to m_i is low (i.e. $T_{j,i} < \theta$). A low transition cost ensures that playing the music from beat m_i to beat m_j and then looping back to m_i sounds natural. In music, longer loops are often less noticeable and therefore we look for the m_i and m_j that maximize $j - i$ subject to the other constraints.

We tune the cost threshold θ to trade-off between finding more loops with higher average cost (high θ) or fewer loops with lower average cost (low θ). We have empirically found that setting θ with respect to the mean and standard deviations of the harmonic and timbre distances provides a good balance:

$$\theta = \frac{\mu_h - \sigma_h}{\sigma_h} + \frac{\mu_t - \sigma_t}{\sigma_t}.$$

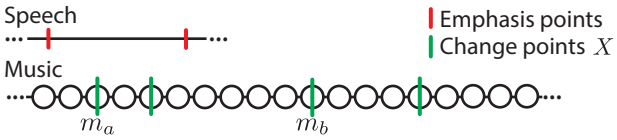

Figure 8. A constrained music retargeting problem in which producer has marked a pair of speech emphasis points (red markers) and selected song. Our system computes the music change points (green markers). We use a dynamic programming approach find a sequence of beats in the music that fit the length of the speech segment such that only the first and last beats change points.

Our interface also allows producers to shorten a segment of music. In this case we search for beats m_i and m_j where $a \leq i < j \leq b$ with $T_{i,j} < \theta$ and $j \neq i+1$. The low transition cost from m_i to m_j ensures that we can safely remove the audio between those beats (Figure 7).

Constrained retargeting

In constrained retargeting the producer wishes to add a continuous musical score to a section of speech containing multiple emphasis points. Moreover, she requires each speech emphasis point to match a change point in the music.

Consider the constrained retargeting problem in Figure 8. The producer has marked a pair of speech emphasis points and selected a piece of music for the score. We automatically find the set of change points X in the selected music using Foote's [18] method for analyzing the *novelty* of each beat based on several audio features including MFCCs, chroma and RMS energy (a measure of volume).

To retarget the music we first compute the number of beats n required to fill the length of time between the speech emphasis points. Note that because the average beat length can vary from song to song we compute n based on the average beat length for the selected song. Our goal is to find a natural sounding sequence of n beats in the music where only the first and last beats are change points. In other words no beat in the sequence other than the first and the last can be in X.

Our algorithm searches for the lowest cost sequence of beats that fits these constraints. We define the cost c of a sequence of n beats starting at beat m_a and ending at beat m_b as $c(m_a, m_b, n)$. To compute $c(m_a, m_b, n)$ we must find the beat $m_k \notin X$ that minimizes the cost of an $n-1$ beat sequence from m_a to m_k plus the cost of a 1 beat sequence from m_k to m_b. That is,

$$c(m_a, m_b, n) = \min \{c(m_a, m_k, n-1) + c(m_k, m_b, 1) \,|\, m_k \notin X\}$$

We set the cost of a 1 beat sequence (i.e. the base case) using the transition cost matrix $c(m_a, m_k, 1) = e^{T_{a,k}}$ so that we favor low-cost transitions. We use dynamic programming to efficiently find the optimal sequence of n beats subject to this recursive cost formulation.

In practice we search for the lowest cost sequence of n beats for each permutation (m_a, m_b) of change points in X. In some cases it is impossible to find a low-cost sequence with exactly n beats. Thus we give the algorithm additional flexibility by allowing the sequence length n to vary by a few beats. We make up for the difference in length by extending or shrinking the pauses that typically follow speech emphasis points in a musical underlay.

Rendering audio

When rendering speech that has been edited, we insert a short crossfade (5 ms long) at each cut to ensure that the cut remains inaudible. Without such crossfades it is sometimes possible to hear faint pops or clicks at a cut. To further improve audio quality we snap edits to zero-crossing points [21], which are the least likely points to cause a click in the generated audio. Our music retargeting algorithms generate a sequence of beats. We render these beats to an audio file by concatenating them in order and inserting a crossfade between any two successive beats m_i and m_j where $j \neq i+1$. These crossfades again serve to hide pops and clicks at loop points in the final rendered audio.

RESULTS

We have used our tools to compose seven audio stories from a variety of raw speech sources including scripted narratives, interviews and political speeches. Table 1 describes the content of each story. All seven final results as well as the original raw recordings can be found on our supplemental website[1] for this paper.

We significantly edited each of the raw recordings to focus the final story on the most important content. The final edited length is usually much shorter than the raw recording length and the total number of cuts indicates the amount of editing we performed. It took us about a half hour to edit each story using our tools. We instrumented our speech and music editing tools to log their usage during each editing session as shown in the table.

As reflected in the usage numbers, we made extensive use of the transcript editor to clean-up and reorganize the speech. Although we used the text delete tool most often, we also used text copy/paste to move words, phrases and sentences. The raw speech recordings for scripted narratives (Elwood and Bullwinkle) often included a large number of re-takes, while interviews and political speeches contained fewer such alternates. When such alternates were available we usually previewed them and selected the best version for the final story. We occasionally had to insert breaths after such rearrangements to maintain the natural rhythm of the speakers.

We added a musical score to all of the stories except for Phone. We used the constrained music retargeting tool to emphasize key moments in the speech for the six other stories. In three of the stories (Photoshop, Obama, LaVette) we also extended some of the music segments by adding loops using our simple retargeting tool.

Limitations. In using our system we also identified two main limitations. First, we found that errors in transcription or in alignment sometimes generated audible artifacts (e.g. clicks, noise, etc.) in the edited speech. We leave it to future work

[1] http://vis.berkeley.edu/papers/audiostories

Name	Description	Raw length	Edited length	Total cuts	Transcript editing usage: Text delete	Text copy/paste	View re-takes	Breath insertion	Music retargeting usage: Simple loops	Constrained retargets
Elwood	Scripted blues show narration	17:11	1:51	14	43	20	5	6	0	1
Bullwinkle	Scripted story about TV show	8:24	1:24	29	29	9	17	6	0	1
Photoshop	Interview about imaging technology	12:22	3:17	64	195	24	2	9	8	3
Obama	Inaugural speech	18:22	2:42	21	61	1	2	0	7	7
Rickard	Interview with math professor	1:07	:54	19	28	1	0	1	0	3
LaVette	Interview with blues singer	8:47	1:45	30	60	18	0	7	19	1
Phone	Lower-quality clip from LaVette	:47	:19	11	41	4	0	7	0	0

Table 1. We constructed seven audio stories using our system and instrumented each of our tools to record usage statistics. Total cuts refers to the number of crossfades that were added by our system when rendering the final audio story. Text delete and text copy/paste refer to usage of basic transcript-based editing tools. View re-takes indicates how many times we previewed different takes of a sentence using similar sentence dropdowns. Breath insertion is the number of times we either added breaths directly or by typing a ('.'). Finally, we counted both the number of loops added to music, and the number of constrained music retargets we used in each editing session.

to provide an interface that allows users to correct inaccuracies in the transcript or in the alignment. Second, we found that in some cases our system was unable to segment a music track into beats (e.g., ambient music with an ambiguous tempo). Poor beat segmentation led to failures in constrained music retargeting. It may be possible to mitigate this problem by adding a music retargeting option that uses constant-sized windows instead of beat-sized windows.

INFORMAL USER EVALUATION

We conducted an informal user evaluation to gauge the utility of our editing interface. We recruited four participants with a range of experience using existing audio editing software: two experts, one casual user, and one novice. We started each session with a 10-minute demonstration of all our editing tools and then asked the participant to create a short audio story from a raw interview recording. At the end of the session, we solicited written qualitative feedback on the features of our interface. In total, each session lasted 50 minutes.

Overall, the results from the study were extremely encouraging. Each participant was able to successfully produce a high-quality audio story that contained significant edits to the speech and music. All of the participants also offered strong positive feedback about the content-based editing capabilities of our interface. One participant wrote, *""This is what narration audio editing should be – it's hard to imagine why I'd want to do it any other way."*

They were most enthusiastic about the transcript-based speech editing tools, which they felt would greatly facilitate the process of editing raw footage into a final story. In particular, they liked the two-column transcript view for linked interview tracks and the ability to quickly modify pauses, breaths, and unnecessary words. One participant said that he thought two-column interview editing was great because *"It made it very easy to see which person's audio I was working on. It's critical to be able to see which speaker is saying what, and I wouldn't get this from a waveform editor (without the tedium/time cost of re-listening to the audio)."*

Participants were also impressed with our music editing tools and felt that our simple and constrained retargeting features allow them to rapidly experiment with musical scoring ideas that would otherwise be prohibitively time-consuming to try out. One participant wrote that constrained music retargeting was *"such a great way to experiment with what might work. Great for trial and error creation."*

The feedback on the music browsing tools was also generally positive. Three of the four participants thought that emotion-based filtering and timbre-based similarity search would be useful. However, we observed that participants spent less time working with the music browser than with our other editing tools. It may be that participants were reluctant to spend too much time searching for music due to the time constraints of the study. Also, one participant noted that he was not as interested in the music browser for this task because he does not personally like audio stories with musical scores.

CONCLUSION AND FUTURE WORK

Audio can be an engaging medium for storytelling. We have presented a set of tools designed to help producers create high-quality audio stories. A key aspect of our tools is that they analyze the content of the raw speech and music tracks to provide higher-level editing capabilities than general-purpose audio editing systems. With our tools producers can focus on building the narrative arc and setting the emotional tone of the story while our system handles the low-level details of editing the audio waveforms.

We believe there are several fruitful directions for building more advanced content-based editing tools. For example it may be useful to apply natural language sentiment analysis [32] techniques to the transcript to identify the most positively or negatively charged sentences. Similarly it may be possible to apply prosody analysis methods [12] to determine which alternate take a sentence is the "angriest". Such analysis could aid a producer in identifying speech emphasis points and in scoring the story based on the emotion of the speech.

Audio stories are typically crafted to fit a certain amount of time. However, some listeners may want to listen to a longer version of the story with more detail, or a shorter, summarized version. Similarly, some listeners may prefer one kind of music in the score while another prefer a different type of music. Future work could investigate how to automatically generate stories of different lengths with personalized scores, from one *master* edit.

ACKNOWLEDGMENTS

We would like to thank Kurt Andersen, Ben Manila, and Julia Wetherell for providing the raw audio recordings we used in this work. We also thank the production staff at WNYC's Studio 360 including Leital Molad, David Krasnow, Michele Siegel, Jenny Lawton, John DeLore, Sean Rameswaram and Alana Harper for showing us how to create compelling audio stories.

REFERENCES

1. Adobe Audition. http://adobe.com/audition, Apr. 2013. Accessed: 2013-04-02.
2. Avid ProTools. http://avid.com/protools, Apr. 2013. Accessed: 2013-04-02.
3. CastingWords. http://castingwords.com/, Apr. 2013. Accessed: 2013-04-02.
4. The EchoNest API. http://developer.echonest.com/docs/v4, Apr. 2013. Accessed: 2013-04-02.
5. Hindenburg Journalist Pro. http://hindenburgsystems.com, Apr. 2013. Accessed: 2013-04-02.
6. Last.fm. http://last.fm/, Apr. 2013. Accessed: 2013-04-02.
7. Abel, J., and Glass, I. *Radio: An Illustrated Guide*. WBEZ Alliance Inc., 1999.
8. Abumrad, J. Music: A force for good (and sometimes evil). http://www.thirdcoastfestival.org/library/450-music-a-force-for-good-and-sometimes-evil, 2005. Accessed: 2013-04-02.
9. Barthet, M., Hargreaves, S., and Sandler, M. Speech/music discrimination in audio podcast using structural segmentation and timbre recognition. *Exploring Music Contents* (2011), 138–162.
10. Berthouzoz, F., Li, W., and Agrawala, M. Tools for placing cuts and transitions in interview video. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings) 31* (2012).
11. Blesser, B. Audio dynamic range compression for minimum perceived distortion. *IEEE Transactions on Audio and Electroacoustics 17*, 1 (1969), 22–32.
12. Boersma, P. Praat, A system for doing phonetics by computer. *Glot International 5*, 9/10 (2002), 341–345.
13. Boll, S. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech and Signal Processing (TASSP) 27*, 2 (1979), 113–120.
14. Casares, J., Long, A. C., Myers, B. A., Bhatnagar, R., Stevens, S. M., Dabbish, L., Yocum, D., and Corbett, A. Simplifying video editing using metadata. *Proceedings of the 4th conference on Designing interactive systems (DIS)* (2002), 157–166.
15. Davis, M. Editing out video editing. *IEEE MultiMedia 10*, 2 (2003), 54–64.
16. Ellis, D. P., and Poliner, G. E. Identifying cover songs with chroma features and dynamic programming beat tracking. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 4* (2007), 1429–1432.
17. Fazekas, G., and Sandler, M. Intelligent editing of studio recordings with the help of automatic music structure extraction. *Audio Engineering Society Convention* (2007).
18. Foote, J. Automatic audio segmentation using a measure of audio novelty. *IEEE International Conference on Multimedia and Expo (ICME) 1* (2000), 452–455.
19. Girgensohn, A., Boreczky, J., Chiu, P., Doherty, J., Foote, J., Golovchinsky, G., Uchihashi, S., and Wilcox, L. A semi-automatic approach to home video editing. *Proceedings of the 13th annual ACM symposium on User interface software and technology (UIST)* (2000), 81–89.
20. Glass, I. The transom review: Ira Glass. http://transom.org/?p=6978, June 2004. Accessed: 2013-04-02.
21. Golding, B. Digital editing basics. http://transom.org/?p=7540, Jan. 2001. Accessed: 2013-04-02.
22. Jehan, T. *Creating music by listening*. PhD thesis, Massachusetts Institute of Technology, 2005.
23. Jehan, T., and Sundram, J. The EchoNest remix earworm. https://github.com/echonest/remix/tree/master/examples/earworm, Apr. 2013. Accessed: 2013-04-02.
24. Kern, J. *Sound reporting: The NPR guide to audio journalism and production*. University of Chicago Press, 2008.
25. Li, F. C., Gupta, A., Sanocki, E., He, L.-w., and Rui, Y. Browsing digital video. *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI) 1*, 6 (2000), 169–176.
26. Lillie, A. S. *MusicBox: Navigating the space of your music*. PhD thesis, Massachusetts Institute of Technology, 2008.
27. Logan, B. Mel frequency cepstral coefficients for music modeling. *International Symposium on Music Information Retrieval (ISMIR)* (2000).
28. Louizou, P. C. *Speech Enhancement: Theory and Practice*. CRC Press, 2007.
29. Lu, L., Wenyin, L., and Zhang, H.-J. Audio textures: Theory and applications. *IEEE Transactions on Speech and Audio Processing 12*, 2 (2004), 156–167.
30. Pachet, F., Aucouturier, J.-J., La Burthe, A., Zils, A., and Beurive, A. The cuidado music browser: an end-to-end electronic music distribution system. *Multimedia Tools and Applications 30*, 3 (2006), 331–349.

31. Pampalk, E., and Goto, M. Musicrainbow: A new user interface to discover artists using audio-based similarity and web-based labeling. *International Conference on Music Information Retrieval (ISMIR)* (2006).

32. Pang, B., and Lee, L. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval 2*, 1-2 (2008), 1–135.

33. Purcell, J. *Dialogue editing for motion pictures: a guide to the invisible art*. Focal Press, 2007.

34. Rubin, S., Berthouzoz, F., Mysore, G. J., Li, W., and Agrawala, M. Underscore: Musical underlays for audio stories. *Proceedings of the 25th ACM Symposium on User Interface Software and Technology (UIST)* (2012), 359–366.

35. Rudnicky, A. Sphinx knowledge base tool. http://www.speech.cs.cmu.edu/tools/lmtool-new.html, Apr. 2013. Accessed: 2013-04-02.

36. Russell, J. A. A circumplex model of affect. *Journal of personality and social psychology 39*, 6 (1980), 1161–1178.

37. Schmidt, E. M., and Kim, Y. E. Modeling musical emotion dynamics with conditional random fields. *International Symposium on Music Information Retrieval (ISMIR)* (2011).

38. Wenner, S., Bazin, J.-C., Sorkine-Hornung, A., Kim, C., and Gross, M. Scalable music: Automatic music retargeting and synthesis. *Computer Graphics Forum (Eurographics conference Proceedings) 32*, 2 (2013), 345–354.

39. Whitman, B. Infinite Jukebox. http://labs.echonest.com/Uploader/index.html, Apr. 2013. Accessed: 2013-04-02.

40. Whittaker, S., and Amento, B. Semantic speech editing. *Proceedings of the SIGCHI conference on Human factors in computing systems 24*, 29 (2004), 527–534.

41. Young, S., Evermann, G., Kershaw, D., Moore, G., Odell, J., Ollason, D., Valtchev, V., and Woodland, P. The HTK book. *Cambridge University Engineering Department* (2002).

42. Yuan, J., and Liberman, M. Speaker identification on the SCOTUS corpus. *Journal of the Acoustical Society of America 123*, 5 (2008), 3878.

43. Zils, A., and Pachet, F. Musical mosaicing. *Digital Audio Effects (DAFx)* (2001).

44. Zolzer, U. *DAFX: Digital Audio Effects*. Wiley Publishing, 2011.