Tools for Creating Audio Stories

Steven Rubin

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-237
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-237.html

December 15, 2015

Acknowledgement

Advisor: Maneesh Agrawala

Tools for Creating Audio Stories

By

Steven Surmacz Rubin

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Maneesh Agrawala, Chair
Professor Björn Hartmann
Professor Greg Niemeyer

Fall 2015

Tools for Creating Audio Stories

Abstract

Tools for Creating Audio Stories

by

Steven Surmacz Rubin

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Maneesh Agrawala, Chair

Audio stories are an engaging form of communication that combines speech and music into compelling narratives. One common production pipeline for creating audio stories involves three main steps: recording speech, editing speech, and editing music. Existing audio recording and editing tools force the story producer to manipulate speech and music tracks via tedious, low-level waveform editing. In contrast, we present tools for each phase of the production pipeline that analyze the audio content of speech and music and thereby allow the producer to work a higher semantic level.

Well-performed audio narrations are a hallmark of captivating podcasts, explainer videos, radio stories, and movie trailers. To record these narrations, professional voiceover actors follow guidelines that describe how to use low-level vocal components—volume, pitch, timbre, and tempo—to deliver performances that emphasize important words while maintaining variety, flow, and diction. Yet, these techniques are not well known outside the professional voiceover community, especially among hobbyist producers looking to create their own narrations. We present *Narration Coach*, an interface that assists novice users in recording scripted narrations. As a user records her narration, our system synchronizes the takes to her script, provides text feedback about how well she is meeting the expert voiceover guidelines, and resynthesizes her recordings to help her hear how she can speak better. In a pilot study, users recorded higher quality narrations using *Narration Coach* than using Adobe Audition, a traditional digital audio workstation (DAW).

Once the producer has captured speech content by recording narrations or interviews, she faces challenges in logging, navigating, and editing the speech. We present a speech editing interface that addresses these challenges. Key features include a transcript-based speech editing tool that automatically propagates edits in the transcript text to the corresponding speech track, and tools that help the producer maintain natural speech cadences by manipulating breaths and pauses. We used this interface 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 tool is easy to learn and greatly facilitates the process of editing raw speech footage into a story.

After the producer edits the speech, she often adds a musical score to the story. We develop an algorithmic framework based on music analysis and dynamic programming optimization that enables automated methods for adding music to audio stories: looping, musical underlays, and emotionally relevant scores. The producer may have a short clip of music that she wants to use in the score; our looping tool allows her to seamlessly extend the clip to her desired length. The producer often uses musical underlays to emphasize key moments in spoken content and give listeners time to reflect on the speech. In a musical underlay, the music fades in to full volume at an emphasis point in the speech. Then the music plays solo for several seconds while the speech pauses. Finally, the music fades out as the speech resumes. At the beginning of the solo, the music often changes in some significant way (e.g. a melody enters or the tempo quickens). Our musical underlay tool automatically finds good candidates for underlays in music tracks, aligns them with the speech, and adjusts their dynamics. Full musical scores often reflect the emotions of the speech throughout the story. We present a system for re-sequencing music tracks to generate emotionally relevant music scores for audio stories. The producer provides a speech track and music tracks and our system gathers emotion labels on the speech through hand-labeling, crowdsourcing, and automatic methods. Evaluations of our looping, underlay, and score generation tools suggest that they can produce high-quality musical scores.

Combined, our tools augment the traditional audio story production pipeline by allowing the producer to create stories using high-level rather than low-level operations on audio clips. Ultimately, we hope that our tools enable the producer to devote more time to storytelling and less time to tedious audio recording and editing.

To Mom, Dad, Len, and Amy.

Contents

List of Figures

List of Tables

Acknowledgments

I started graduate school by working on an information visualization project with Maneesh Agrawala. But in October of my first semester, when he learned that I had an interest in audio, Maneesh began talking with me about how he wanted to improve radio production. Over the next four years, Maneesh and I, with our collaborators Floraine Berthouzoz, Gautham Mysore, and Wilmot Li, built audio story creation tools based on ideas that all trace back to those initial conversations. As my PhD application essays corroborate, I did not have a clear research agenda for graduate school. I entered Berkeley as an Operating Systems student, but Maneesh's work excited me. I thank him for his commitment and mentorship. He supported me at all stages of the process, from shepherding me through my first readings of the HCI and infovis "classics," to having stimulating brainstorming sessions and research debates, to motivating me during the eighty-hour workweeks before paper deadlines. Maneesh knew exactly how and when to back off as I gradually figured out things for myself. His support and dedication were essential to my rewarding graduate career. I also thank my other committee members, Björn Hartmann and Greg Niemeyer, for their helpful feedback on this dissertation.

I was fortunate to develop close, long-term collaborations with Gautham, Wil, and Floraine at Adobe Research. Always quick to offer the help and advice I needed, Gautham, Wil, and Floraine were fantastic collaborators and mentors. Floraine tragically passed away in August of 2015. She made her academic career in seeking out game-changing projects in HCI and computer graphics. Not only was she my role model at Berkeley as I started graduate school, but she has been an inspiring collaborator and friend over the past four years. I miss her.

My friends in and out of lab made the years at Berkeley fly by. I thank lab-mates Sean Arietta, Peggy Chi, Alex Hall, Shiry Ginosar, Jonathan Harper, Jessica Hullman, Kenrick Kin, Colorado Reed, Armin Samii, Valkyrie Savage, Wes Willett, and Eric Yao for their lively conversations. Outside of lab, my friends Sarah Anne Bender, Camille Chicklis, Rachel and Rusty Cowher, Danielle Diuguid, Marissa Pilger, David Roth, Julian Suhr, and Dan Waters made sure I scaled plenty of fake rocks, basked in the Golden Age of Television, and sampled the countless gastronomic delights of the East Bay. Amy Pavel is the intersection of my work/play Venn diagram. Since nearly the beginning of graduate school, she has been my constant companion. She has shared with me her passions for travel, photography, and the outdoors. I have shared with her my snobbery for music, board games, and coffee. We're a good team.

Finally, I acknowledge my family—Scott and Cindy, and my brother Len. I can't adequately summarize their positive impact on my life, so suffice it to say: I thank them for everything.

Chapter 1

Introduction

Audio stories—ranging from audio books to radio stories and podcasts—combine speech and music into compelling, engaging narratives. New technology is continually simplifying the process of capturing and distributing multimedia content. With the increased fidelity and quantity of this media, a question arises: *how can everyday users produce high-quality media?* Users tell others about their lives by sharing raw, unedited media, but these exchanges are not structured as stories, and they are not edited to resemble the high-quality productions we hear on the radio or highly produced podcasts.

Although social sharing buttons in smartphones and desktop applications encourage an unfiltered share-it-all mentality, users have more structured stories to tell. But how can they turn their recordings into a story that could appear on *This American Life*? Expert-created content is often more enticing—the stories are well-paced, thoughtfully structured, and augmented with emotionally-fitting transitions and musical scores—but the editing required in its creation is both challenging and time-consuming. While existing content creation tools provide expert users near-unlimited flexibility and power, they remain daunting and difficult to learn for the average user [46].

Our goal is to improve the audio storytelling process by creating media-specific algorithms for new recording and editing user interfaces. We aim to amplify creativity by giving users of all skill-levels powerful and flexible tools while avoiding creating complicated interfaces. Yet we also want to avoid interfaces that are too simplistic and may stifle creativity. Our tools allow users to record and edit audio at a high level instead of with low-level waveforms. The coupling of algorithms and insights from the audio/music information retrieval communities, along with an understanding of design guidelines for audio storytelling result in tools that both expert *and* amateur content-creators can use to produce compelling, high-quality stories.

1.1 Audio story production

In order to create a compelling audio story, a producer first needs to have a good story to tell. This stage of the production process involves brainstorming, researching, writing, and potentially interviewing people related to the story. Even if its underlying story is new and interesting, an audio story

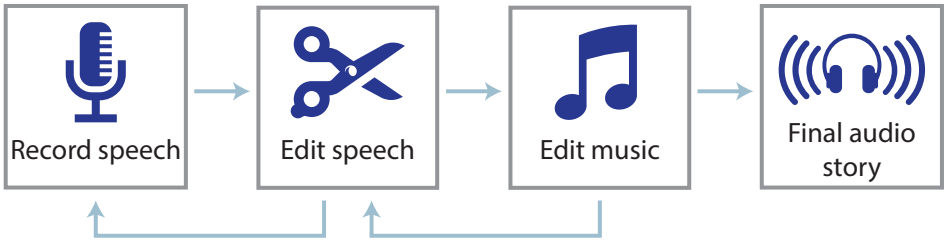

Figure 1.1: In our research, we develop tools to assist producers in creating high-quality audio stories. These tools address problems in recording speech, editing speech, and editing musical scores for audio stories. This is an iterative pipeline, e.g., the producer may decide to re-record narration after working on an initial music edit.

may fail to capture a listener's attention if the story is poorly produced. Our research focuses on the production process that occurs after this initial ideation phase. Figure 1.1 shows one characteristic pipeline for this production process. This iterative pipeline breaks the production process into three main challenges: recording speech, editing speech, and editing music.

1.1.1 Recording speech

In high-quality narrations, professional voiceover actors adjust the volume, pitch, timbre, and tempo of their voices to follow a set of best practices. They emphasize important words, vary the tempo and pitch of the delivery, control the speed of the delivery, and articulate words clearly. How can a novice user follow these guidelines to record a high-quality narration?

1.1.2 Editing speech

The producer needs to take a potentially large collection of raw speech narrations and interviews and edit them to form her story. How can the producer navigate and edit raw speech recordings without having to tediously annotate the recordings or manipulate them as low-level signals? How can she perform more advanced speech editing techniques like maintaining breaths and pauses at appropriate locations?

1.1.3 Editing music

Most audio stories include musical scores that add texture and meaning to the speech. How can the producer effectively incorporate music into a story? How can she adjust the positioning and dynamics of the speech and music to best emphasize parts of the story? How can she create a musical score that matches and highlights the emotions of the speech? Some audio story producers compose and record their own music; however, we focus our research on the challenges in editing existing music to fit stories.

1.2 New production tools

In our research we design and build tools that address these challenges, thus creating an improved pipeline for amateur and expert producers to create audio stories. In developing new audio production tools, we follow the approach of Agrawala et al. [7] and first identify a set of design guidelines for each task. We study publications that describe how to record effective narrations [45] and edit them together with music [2, 4, 5, 12, 44] to create high-quality audio stories. In addition, we listen to and analyze hours of highly produced radio programs to further understand how expert producers apply these guidelines in practice. As a guiding principle, our tools aim to provide functionality at a higher semantic level than traditional recording and editing tools. For example, our research enables the producer to record narration, edit speech, and construct musical scores without requiring her to directly manipulate waveforms.

We present background information and research related to our new tools in Chapter 2 (Related work). In Chapter 3 (Recording speech) we describe a tool [78] that helps amateur voiceover actors record higher quality narrations by providing them feedback about how well they are adhering to high-level narration guidelines. Chapter 4 (Editing speech) details our editing interface [77] that allows the producer to edit speech audio using text rather than waveforms, and helps them fine-tune breaths and pauses in speech. In Chapter 5 (Editing music) we define an algorithmic framework for simplifying the creation of compelling musical scores for audio stories [75, 76, 77]. This includes tools for looping music, emphasizing key moments, and automatically creating full-length, emotionally relevant musical scores that highlight the emotions of the speech. Finally, in Chapter 6 (Integrity of manipulated audio) we discuss the implications of tools designed to manipulate media like speech and music. We outline the ways that our recording and editing tools help the producer understand and track her changes to the source material.

This dissertation is based on papers published in ACM conference proceedings with coauthors Maneesh Agrawala, Floraine Berthouzoz, Gautham Mysore, and Wilmot Li. I am the primary author on all of these publications. The work on recording speech in Chapter 3 was published in UIST 2015 [78]; the speech editing and music editing interface and algorithms in Chapters 4 and 5 were published in UIST 2013 [77]; our investigation of musical underlays and emotionally relevant musical scores in Chapter 5 were published in UIST 2012 [76] and 2014 [75], respectively.

Chapter 2

Related work

Related research on improving the audio story creation process is sparse. Our research takes inspiration from work on active capture user interfaces, audio/speech signal processing, text-to-speech systems, music information retrieval, and concatenative synthesis.

2.1 Recording speech

Our work on recording speech builds on research in active capture systems and audio resynthesis methods.

2.1.1 Active capture

Narration Coach follows in a line of research on *active capture* [26], a media-production paradigm that combines capture, interaction, and processing. Chang and Davis [24] present an active capture system to direct users performing and recording a video introduction. Heer et al. [47] describe strategies for active capture systems to deliver feedback to users. Our system differs from these projects by providing an end-to-end system for the broad task of recording narration rather than scene-specific directions like "turn your head" and "scream louder." Carter et al.'s *NudgeCam* [21] helps users record video that follows capture heuristics such as interview guidelines. *Narration Coach* similarly helps users follow capture heuristics, but for audio narration instead of video. Hindenburg Audio Book Creator [48] is a digital audio workstation specifically designed for recording audiobook narration. It allows users to manually link recordings to a script, but unlike *Narration Coach*, it does provide feedback and resynthesis tools.

The research most similar to our work on speech recording is Kurihara et al.'s *Presentation Sensei* [57], a tool that uses speech and image processing to give a speaker feedback on speed, pitch, and eye contact while he rehearses a slide-based presentation. Like our system, *Presentation Sensei* provides capture-time feedback about speech performance. However, our system focuses on the iterative narration recording process. It organizes recorded audio based on an input script, and it provides automatic, word-level feedback and resynthesis tools.

2.1.2 Resynthesizing audio recordings

Existing digital audio workstations (DAWs) like Avid ProTools and Adobe Audition allow users to record audio and improve its quality using low-level signal processing effects. However, these DAWs target professional audio producers and unlike our system, they do not provide high-level tools to help novices record narrations, like associating recordings with lines in a script and providing automated feedback and resynthesis.

Our work draws on techniques from digital signal processing research to analyze and manipulate speech. Researchers have developed speech prosody manipulation techniques such as the phase vocoder [31], which enables audio adjustments by modifying components in the frequency domain, PSOLA [93], which reconstructs audio by adding, removing, and shifting small windows in the audio, and TANDEM-STRAIGHT [54], a vocoder that allows manipulations by decomposing speech into a smooth spectrogram, pitch contour, and periodicity map. Black and Taylor describe the Festival system [14, 92]—a complete text-to-speech system—which include concepts relevant to our work such as prosody prediction from text, prosody synthesis, and speech signal analysis.

2.2 Editing speech

2.2.1 Traditional tools

Just as traditional DAWs provide users precise tools to record audio, they also allow users to edit and manipulate audio at the signal level. They are general-purpose audio production systems whose features are not directly targeted at creating audio stories. In addition to Audio Book Creator, Hindenburg Systems [1] develops tools 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 speech waveforms.

2.2.2 Alternatives to waveform navigation and editing

Researchers have investigated 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. [37] 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 [27] 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. [60] enable faster browsing of video clips by automatically marking shot boundaries and enabling pitch-preserving rapid audio playback. Likewise, Girgensohn et al. [42] present a system that automatically filters out unsuitable video clips based on an analysis of the camera motion. Unlike these

systems, our speech editing interface focuses on using a transcript of the speech track to navigate and edit audio stories.

2.2.3 Text-based speech editing

We build on previous techniques for text-based navigation and editing. Whittaker and Amento [98] 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. [22] and more recently, Berthouzoz et al. [11] 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 domain of audio editing and provide a number of additional transcript-based editing tools (e.g. grouping similar sentences and editing breaths).

2.3 Editing music

2.3.1 Automatically scoring stories

Automatically scoring audio and video stories is a longstanding problem in multimedia research. Foote et al. [41] automatically create music videos by editing video clips to match a piece of music. Their method finds suitable locations to cut video and then splices video to match points in music where audio features change. Our work on scoring stories instead edits the music to create a soundtrack tailored to the speech by matching emotions and speech emphasis points rather than low-level audio/video features. Monteith et al. [67, 66] generate new music MIDIs to match automatically labeled emotions in stories. They learn generative models of different music emotions and then sample new music from these models. However, because their algorithm only considers local matching of speech and music, the soundtracks they generate may not contain story-level structure or cohesion. Our full score retargeting algorithm uses the emotion labels across an entire story to generate a globally coherent musical score. We generate musical underlays and emotionally relevant musical scores by re-sequencing existing, high quality music tracks instead of MIDIs, and we provide more options for emotion labeling.

2.3.2 Concatenative synthesis

Example-based music synthesis and retargeting is an active area of research [51]. Lu et al. [62] 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 [105] Musical Mosaicing system synthesizes a target waveform by sequencing snippets from a database of music. The EchoNest, a subsidiary of Spotify 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 [52, 97]. In concurrent work, Wenner et al. [96] 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.

Dynamic programming is a common strategy for concatenative music synthesis, where short music clips are re-sequenced from a database to create new music that mimics existing music, or to create music that follows certain note patterns [86, 85] Our work on music retargeting follows this general approach, but we offer constraints that generate musical underlays and emotionally relevant music scores rather than music that imitates other music. Other work applies belief propagation [91] and genetic algorithms [95] to music re-sequencing, but we use dynamic programming instead to efficiently find optimal solutions. Algorithms for video synthesis often apply dynamic programming techniques as well, as in Schödl et al.'s [84] video textures, and Arikan et al.'s [8] annotation-based character motion synthesis.

2.3.3 Music editing interfaces

Previous research has investigated user interfaces and interaction techniques that simplify music editing workflows. Much of this research focuses on the task of creating music and proposes graphical representations [36, 101] and tangible multitouch interfaces [10, 13, 19, 73] that support music composition and live performance. Other researchers have developed tools that leverage musical structure and metadata to enable users to edit music at a semantically meaningful level such as mixing and matching multiple takes from a music recording session [25, 37] rather than working directly with waveforms or spectrograms. There has also been previous work that supports vocalized user input for selecting specific sounds in audio mixtures [90] and for creating musical accompaniments to vocal melodies [89]. Most of this research adopts the general strategy of identifying the requirements and constraints of specific music editing tasks in order to design user interfaces that expose only the most relevant parameters or interaction methods to the user. We follow a similar strategy for the task of creating and retargeting musical scores.

2.3.4 Affect prediction

Affect prediction is a well-studied problem for speech [34], text [18], and music [56]. We draw on these techniques to provide automatic emotion labeling tools in our system. Our work concentrates on the application of affect labeling and understanding rather than on its prediction.

Chapter 3

Recording speech

From podcasts and radio stories to explainer videos to commercials and movie trailers, audio narration pervades modern media. High-quality narrations captivate listeners and shape their perceptions of stories. But producing an effective narration requires good delivery. Professional voiceover actors continuously adjust four primary voice components—volume, pitch, timbre, and tempo—to follow a set of best practices for high-quality audio narrations. These best practices cover four high-level guidelines [45]:

- **Emphasis:** Emphasize important words by adjusting the voice's pitch contour, volume and tempo.
- **Variety:** Vary the tempo and pitch of the delivery to avoid sounding monotonic.
- **Flow:** Control the speed of the narration and avoid pauses between words to allow sentences to flow naturally.
- **Diction:** Articulate words clearly; do not mumble.

Hobbyist content creators also record narrations, but are not always aware of these professional voiceover guidelines. As a result, their voiceovers often sound less captivating, coherent, and polished. Even a novice creator who knows that she should, for example, emphasize the important words in a sentence, may struggle in manipulating the pitch of her voice to achieve that emphasis.

In this paper we present *Narration Coach*, an interface that assists novice users while they record scripted narrations by providing immediate feedback on how well the user emphasizes important words and maintains variety, natural flow and good diction. The user enters her desired script and underlines words that she wants to emphasize. As the user records herself speaking the script, our system segments and aligns the recordings to the corresponding lines in the script. After the user records a set of lines, *Narration Coach* detects the spoken emphasis in those lines, and checks for vocal variety, good flow, and clear diction in the recording. Our system uses these detections in two ways. First, it provides feedback to the user about how successfully she spoke the line and what she can improve. Second, it provides methods for resynthesizing versions of the recording that improve the emphasis and flow. The user can construct her final narration either by re-recording lines with

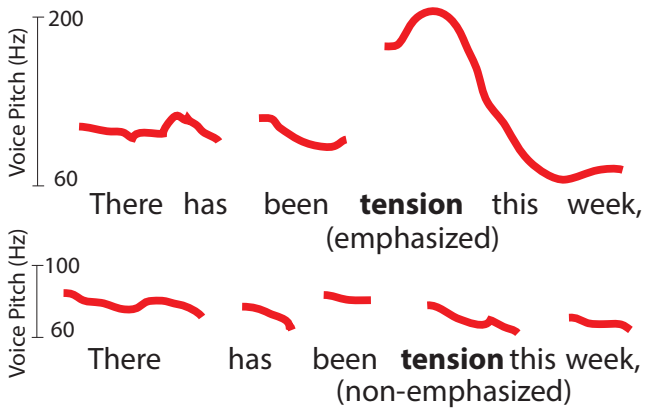

Figure 3.1: In two spoken versions of the same phrase, the pitch contour for the word "tension" features a steep pitch increase when the speaker emphasizes the word (top) versus when she does not (bottom).

the feedback in mind, or by using one of the resynthesized options as an improved version of the take. This record–feedback–resynthesis pipeline allows users to iteratively improve their narration.

In a pilot study, first-time users of our system successfully created audio narrations that they preferred over narrations they made without *Narration Coach*. Additionally, impartial listeners rated narrations created with our system as higher quality than the narration made without it.

3.1 Guidelines for high-quality narration

We designed *Narration Coach* to address common problems in recording narration [28, 32, 45, 71]. Voiceover actors use four high-level guidelines to improve the quality of their delivery [45, 72]. They emphasize words that help users follow the story (e.g. descriptive words, proper nouns, action verbs), add vocal variations to the delivery, adjust the speed of the narration and the location of pauses to control the flow of the speech, and enunciate words clearly. To achieve these high-level goals, voiceover actors continuously adjust four components of their voice, i.e. the volume, pitch, timbre, and tempo.

3.1.1 Emphasis

When speaking, voiceover actors adjust their pitch, tempo and volume to emphasize or *hit* a word. Emphasizing a word makes it stand out or signals the end of a thought; it helps the listener follow the story and identify the most important messages. The speaker needs to understand the intended emotion and meaning of the narration to determine which words to emphasize. However, general-purpose voiceover guidelines suggest: emphasizing one of the first two words in a sentence can help listeners focus attention on the remainder of the sentence; emphasizing the end of a sentence signals the end of a thought; emphasizing action words and descriptive words helps listeners focus on the

action and the subject; and emphasizing reference words (e.g. subject names) helps listeners focus on the subject, location and objects in the story [45].

Voiceover acting books further suggest that the most common way to emphasize a word is to increase the pitch on the stressed syllable of the word while slightly increasing the volume and duration of the word [15, 45, 99]. A common emphasis pitch contour of a word starts at a low pitch, rises to a greater pitch at the stressed syllable in the word, and then falls back down to a lower pitch by the end of the word (Figure 3.1). We can provide feedback on whether the speaker emphasized each word by analyzing its pitch contour, volume, and duration. To resynthesize a sentence to emphasize a non-emphasized word, our system adjusts the pitch contour of the word to mimic a target emphasis contour, and increases the duration and volume to draw further attention to the word.

3.1.2 Variety

If a speaker uses a small range of pitches and always speaks at the same tempo, the resulting speech can sound robotic and listeners may lose focus. Instead, to keep the voiceover interesting, speakers add variety to their delivery. They use pitch variation to expand their range, and elongate words to change the tempo of the sentence. We analyze the pitch and tempo to provide feedback about the variety in the recording. Our system focuses on helping users speak with more variety. Although some speakers have *too much* vocal variety, novice speakers are less likely to have this issue.

3.1.3 Flow

To control the flow of a sentence, voice actors continuously adjust the tempo of the narration, so as not to speak too fast or too slow. They are also careful about where they insert pauses to avoid unnecessarily disrupting a sentence. Less experienced speakers often read in a disconnected way; they insert extraneous pauses between words or speak so fast that they forget to pause entirely. Speakers can improve the flow of a narration by speaking at a natural tempo and minimizing unintentional pauses. We provide feedback about the flow by analyzing the speed of the narration, i.e. by comparing the duration of each spoken word to the typical duration of that word (learned from data). We also analyze the duration and location of pauses within a sentence. To resynthesize the recording with improved flow, we can slow down or speed up the tempo, as well as insert or remove pauses as necessary.

3.1.4 Diction

To make voiceovers easy to understand and follow, speakers must articulate their words clearly. Clear enunciation correlates with exaggerated mouth movements [72]. We thus provide feedback about the quality of the diction by using a face tracker to analyze the speaker's mouth movements throughout a recording session. We do not resynthesize speech to correct for poor diction. Increasing or decreasing enunciation in recorded speech is an open problem for future research.

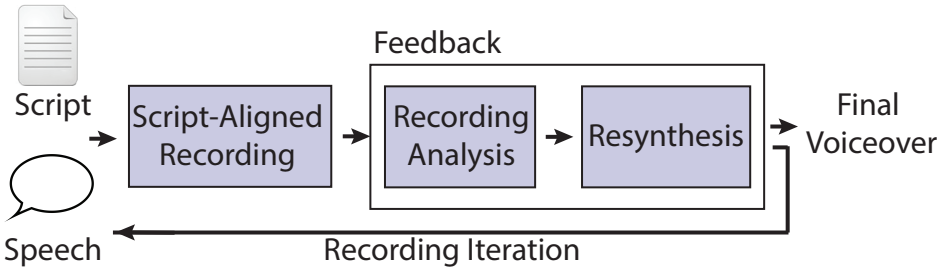

Figure 3.2: A user records part of a narration script in our system. After she records part of the script, our system displays text-based feedback and provides audio resyntheses that correct problems in the narration. This text and audio feedback helps her understand how to iterate and improve the recording.

3.2 Interface

This section describes our system's interface, while Section 3.3 explains the techniques that enable these interactions. At a high-level, the user records part of the script and *Narration Coach* continuously time-aligns the recording with the script (Figure 3.2). When the user stops recording, the interface provides feedback for each recorded take based on the four high-level narration guidelines. *Narration Coach* also resynthesizes the recorded speech to automatically improve it. The user can incorporate the resynthesized version in their final narration, or use it as a suggestion of how to improve their narration. In our system, we call a sentence in the script text a ***line***, and we call an audio recording of one line a ***take*** of that line.

3.2.1 Transcript-guided recording

The user launches *Narration Coach* after she writes her script. She imports the script text file and our interface shows a line-by-line view of the script in the main window (Figure 3.3, left), where each line corresponds to a sentence. She can edit, add, and remove lines from the script in this window. She can also select words and mark them as targets for emphasis (⌘+U). Our interface underlines such emphasis words.

If the user is unsure about which words to emphasize, she can select a line and click on the "Annotate line" button in the toolbar, which underlines suggested words to emphasize. *Narration Coach* does not annotate the entire script automatically by default because desired emphasis can be context-dependent; a speaker may emphasize the same sentence in different ways to convey different meanings. Nevertheless, if the user does not have a specific interpretation in mind, our system applies a general set of rules as described in Section 3.1 to pick words to emphasize.

To begin recording the script, the user clicks the "Record" button on the toolbar and starts reading the script line by line (Figure 3.3a). She does not need to read the lines exactly as they appear in the script; the speaker can modify lines, e.g., by re-wording phrases.

When the user presses the "stop recording" button, *Narration Coach* analyzes the speech. It first

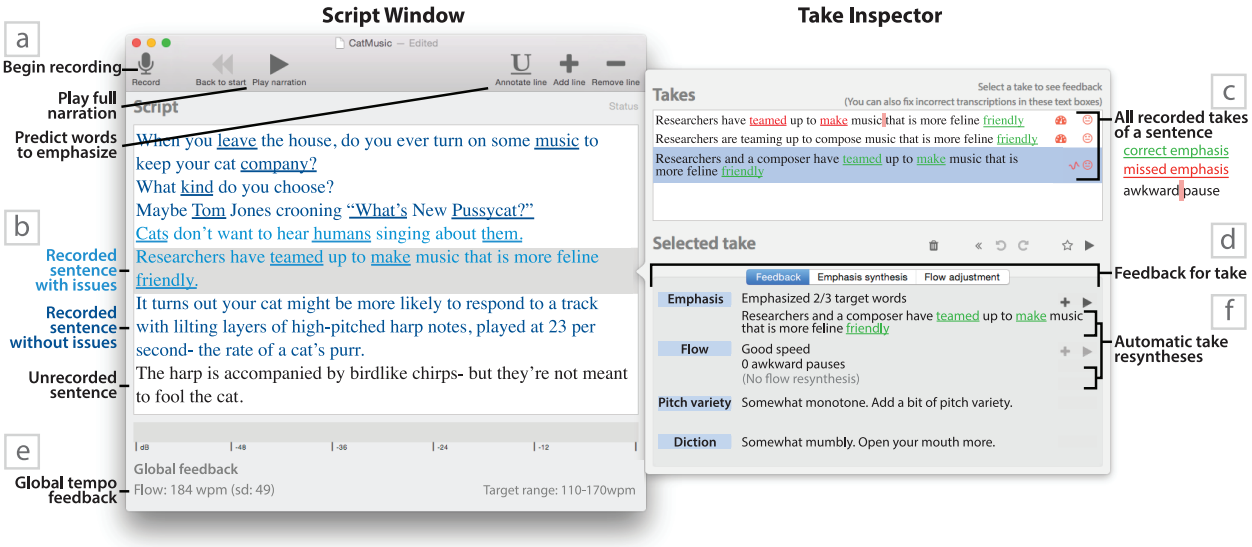

Figure 3.3: The main window of our interface shows the narration script (left). The user (a) underlines words that she intends to emphasize and records part of the script. The script window (b) uses font color to show which lines the user has not recorded yet, which lines have problematic recordings, and which lines have good recordings. When the user clicks on a blue line in the script, our system displays the take inspector (right) for recordings of that line. The take inspector (c) shows the user words she correctly emphasized and words she failed to emphasize. By selecting a specific take of the line, she can (d) see feedback about that take's emphasis, flow, variety, and diction. Our system also shows (e) global tempo variety. She can (f) listen to resynthesized versions of the recording that address emphasis and flow issues and replace the take with a resynthesized version.

transcribes the speech and then automatically segments and aligns the speech transcription to the best matching set of consecutive lines in the script. The main script view displays these recorded lines in a light blue font to indicate to the user that those lines have recorded takes associated with them (Figure 3.3b). If after the speech analysis, our system determines that a recorded line follows all four high-level narration guidelines, the script view instead colors the line in dark blue. The font colors let the user quickly see which parts of the script she has yet to record (black lines), and which parts to record again to improve the quality of the narration (light blue lines). When the user clicks a blue-colored line in the script, the *take inspector* appears and shows all of the recorded takes for that line (Figure 3.3, right). The take inspector also provides feedback about the delivery and allows the user to resynthesize takes.

3.2.2 Speech feedback

In the take inspector (Figure 3.3, right), *Narration Coach* provides feedback about the four high-level guidelines: emphasis, variety, flow, and diction. *Narration Coach* detects the emphasized words in each recorded take. As in the script view, the take inspector underlines each word that the user intended to emphasize. For each of these words, the take inspector renders the word in green if the user successfully emphasized the word and red if the user did not emphasize the word (Figure 3.3c). This visual encoding allows the user to see where she needs to place more emphasis in subsequent recordings. If the user disagrees with the detected emphasis, she can select words in the recorded take and add or remove the emphasis (⌘+B).

For other guidelines, we aim to give the user actionable advice rather than have her focus on specific quantitative properties of her speech. To do this, we provide text-based feedback rather than solely giving numerical or visualization-based feedback.

Narration Coach provides textual feedback describing the variety of the performance. Feedback lines indicate the variety in the pitch and tempo of the speech and suggest that the user add more or less variety in further recordings if needed. Our system reports pitch variety feedback for each recorded take (Figure 3.3d). While excessive *tempo* variety within a single line sounds unnatural, tempo variety over multiple sentences helps to hold the listener's attention. Our system gives tempo variety feedback about the full narration in the script window (Figure 3.3e).

To provide feedback about the flow of the speech, our system displays the speech tempo and the number of mid-sentence silences. These displays suggest that the user slows down or speeds up, and reduces unnecessary mid-sentence pauses. *Narration Coach* renders awkward pauses as red boxes in the text of the take. We also use text to provide feedback about the quality of the diction, suggesting that a user open her mouth more to improve mumbled speech (see Section 3.3 for details on text feedback).

3.2.3 Speech resynthesis

Resynthesizing audio—splicing multiple takes together or manipulating the underlying pitch, volume, and duration contours (time-varying functions) of individual takes—allows the user to create and hear modified versions of her performance. Our system automatically generates resynthesized

takes to address problems with emphasis and flow. As our system resynthesizes audio with contours further from the original take's contours, the resulting audio becomes more audibly different from the original recording, but also gains more audio artifacts. The user can use these resynthesized takes to replace her original take, or she can use them as audio feedback to guide her performance next time she records a take of that line.

Narration Coach provides two forms of speech resynthesis: emphasis and flow (Figure 3.3f). When the user selects a sentence in the take inspector, our system automatically generates these two resynthesized versions of the sentence. If the user fails to emphasize words that she underlined in the script, our system generates a version of the sentence that adds emphasis to those words. The user can click on the "plus" button to replace the original take with the resynthesized take.

If the user spoke too quickly or too slowly in the recording, *Narration Coach* resynthesizes a slower or faster version of the take, respectively. This resynthesized take adjusts the speed while preserving the other characteristics of the recording. Our system also automatically adds pauses between sentences and reduces unnecessary pauses within a sentence.

3.2.4 Constructing the final narration

As the user records and re-records the lines in her script, *Narration Coach* captures a complete recording of the script. The user can listen to this recording at any point by clicking the "play narration" button in the main script window. Our system analyzes the problems in each line to automatically select and play the best take of each line for the final version of the voiceover. The user can override this default "best-take" selection behavior by clicking the "star" button on her favorite take in the take inspector. When the user finishes recording the narration, she selects "Export" from the file menu to export her full narration as a high quality, uncompressed WAV file.

3.3 Algorithmic methods

The features in *Narration Coach* rely on text and audio processing algorithms.

3.3.1 Script Analysis

Narration Coach provides suggestions for words in the script to emphasize by applying rules proposed by voiceover experts [45]. Our system suggests users emphasize: descriptive words (adjectives), proper nouns, action verbs, and words at the beginnings and ends of sentences. *Narration Coach* uses the Mac OSX built-in `NSLinguisticTagger` API to tag the part-of-speech of each word, and then applies the rules listed in Section 3.1 to label emphasis. The tagger does not differentiate between action verbs and non-action verbs, so we supply a list of non-action verbs to ignore during annotation.

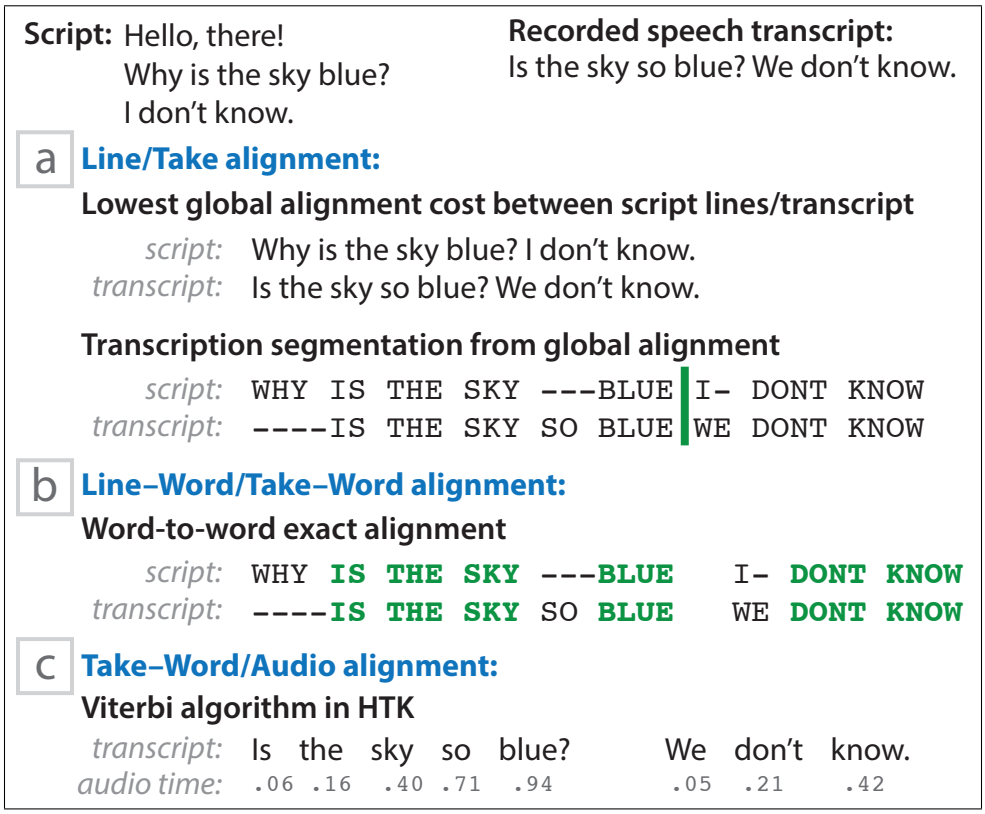

Figure 3.4: *Narration Coach* aligns recorded speech with the script. First, (a) it finds the set of script lines that correspond to the recording. Then, it segments the transcript based on a global alignment with the script lines. It (b) finds word matches between the script lines and the transcript. Finally, (c) it aligns the words in the take to the speech audio.

3.3.2 Transcript-guided recording

While the user is speaking, our system records the audio and runs Mac OSX's Enhanced Dictation tool in the background to compute a text transcript. In order for our interface to organize, analyze, and manipulate the recorded audio, it requires alignment metadata (Figure 3.4). When the user stops recording, our system performs a multi-staged alignment process to generate the following metadata:

- The **line/take alignment** is the link between each recorded take and a line in the original script, so our system can show users all takes of a given script line.
- The **line-word/take-word alignment** links words in each recorded take and words in the line corresponding to that take. If the user did not read the script verbatim, some words in the recorded take may not have links to the original line, and some words in the original line may not have links to the recorded take. We need this metadata to determine if the take includes the words that the user intended to emphasize.

- Finally, the **take-word/audio alignment** is a set of timestamps in a take's recorded audio for the beginning and end of each word and phoneme, so our system can analyze and manipulate the vocal components of the recording at the word and phoneme levels.

Our alignment process makes no assumption about where the user starts speaking within the script, but does assume that she speaks lines sequentially from the starting point. The user also does not have to speak lines exactly as written— she can change wordings while she is recording. Because the user can record multiple script lines at once, our system first segments the transcript t into its associated script lines. The transcript corresponds to a consecutive interval $[i, j]$ of script lines $\mathbf{s}$. Our system finds the interval with the best match to the transcript, as measured by the minimum cost global alignment $c(t, s_{[i,j]})$ between the transcript and the concatenated script lines $s_{[i,j]} = s_i + \cdots + s_j$. We find this interval by first applying Needleman-Wunsch [68] to compute the global alignment between the transcript t and each script line s_i. The cost of this alignment corresponds to a score for interval $[i, i]$. Then, our system repeatedly extends each of these intervals by one line as long as the alignment cost decreases, i.e., $c(t, s_{[i,j+1]}) < c(t, s_{[i,j]})$. From these final intervals, our system selects the one with the smallest cost. Our system then segments t into $j - i$ sentences according to this minimum cost alignment, which gives us the line/take alignment (Figure 3.4a).

Our system finds the line-word/take-word alignment by searching the global alignment from the previous step for words in the transcript that are exactly aligned to words in the script (Figure 3.4b). Finally, *Narration Coach* computes word and phoneme timestamps for the recorded audio using the HVite component of HTK, a hidden Markov model toolkit [102] (Figure 3.4c).

3.3.3 Speech feedback

To detect emphasized words, *Narration Coach* takes a two-phased approach. AuToBI [74]—a tool for predicting prosodic annotations in speech—reliably detects words that have pitch accents, which are pitch "configurations that lend prominence to their associated word" [88]. While pitch accents are necessary for emphasis, they are not sufficient. For example, a word preceding a pitch accented word may contain a much larger pitch accent, overpowering any emphasis that a listener may otherwise hear in the second word. We apply the AuToBI pitch accent detector to find a subset of words that have pitch accents. In order to find the pitch-accented words that sound emphasized, our system searches for typical emphasis contours (Figure 3.1): our system runs a second pass over the pitch-accented words, finding those that are louder (> 1.25 decibels) or higher pitched (> 1.25 times) than the preceding word, or those that have more pitch variation (> 1 standard deviation) than the sentence as whole. We set these thresholds by analyzing contours before, during, and after emphasized words in speech recordings. *Narration Coach* underlines the user's target emphasized words, rendering successfully hit target words in green and missed target words in red.

Our system finds pitch variety by first applying Mauch and Dixon's [63] probabilistic YIN smooth pitch estimation algorithm and computing the standard deviation of the log of the pitch in the take. *Narration Coach* displays this standard deviation in the take inspector, mapping the value to a level of text feedback— $[0, .2)$: "Monotone. Add more pitch variety," $[.2, .3)$: "Somewhat monotone, Add a bit of pitch variety," $[.3, .4)$: "Good pitch variety," and $[.4, \infty)$: "Excellent pitch variety." We set

these mappings empirically by listening to and qualitatively rating narration audio clips. If the user finds the mapping to be inaccurate, she can adjust the mapping through our system's preferences window.

We provide feedback on tempo variety for the narration as a whole. *Narration Coach* computes the words per minute (WPM) and the standard deviation of a moving average of WPM of the full narration and displays these values below the script in the main script window.

Our system uses a different tempo metric to provide flow feedback at the take-level. Sentences have large differences in word length distribution, so WPM is less useful as a metric for take-level speed analysis. Instead, our system computes the speed of each take by comparing the duration of each spoken word to the expected duration of that word. We use the transition probabilities in a monophonic hidden Markov model [104] to model the duration of each English phoneme. The expected duration of a word is the sum of the expected durations of the phonemes in that word. Our system computes the cumulative distribution function (CDF) for the duration of the words in each take to report a value between zero—the speaker said the sentence as quickly as possible—and 1—the speaker said the sentence as slowly as possible. *Narration Coach* maps the probability to different levels of text feedback— [0, .25): "Very slow! Speed up," [.25, .3): "Somewhat slow. Speed up," [.3, .4): "Good speed," [.4, .45): "Somewhat fast. Slow down," and [.45, 1]: "Very fast! Slow down." As with the pitch variety mappings, we set these empirically and they are user-customizable.

When people articulate clearly, they open their mouths wider than when they mumble [72]. In addition to recording audio, *Narration Coach* captures video from the computer's webcam as the user speaks. Our system detects mumbling by analyzing the speaker's facial movement. *Narration Coach* runs Saragih's face tracker [81] on the captured video and records four points for each captured frame: the topmost mouth point, the bottommost mouth point, the topmost eye point, and the bottommost nose point (see Figure 3.5). Our system computes the ratio $\frac{mouth_{top} - mouth_{bottom}}{eyes_{top} - nose_{bottom}}$ for each frame and then computes the standard deviation of these ratios. The higher the variance of the $mouth_{top} - mouth_{bottom}$ difference, the wider the speaker is opening her mouth while speaking. The denominator acts as a normalizing term to preserve scale-invariance in this ratio. *Narration Coach* maps the standard deviation of these ratios to text feedback in the take inspector— [0, .02): "Very mumbly! Open your mouth more," [.02, .04): "Somewhat mumbly. Open your mouth more," and [.04, ∞): "Well-articulated." Before the user starts recording, we calibrate these mappings based on the size of the user's closed mouth and the size of the user's wide-open mouth.

3.3.4 Speech resynthesis

Narration Coach provides automatic speech resynthesis methods for emphasis and for flow. If the user fails to emphasize words she underlined in the script, *Narration Coach* resynthesizes the take with emphasis added to those words in a two-phased procedure. First, for each missed target word, our system checks if the user properly emphasized the word in another take of the same script line. If she did, *Narration Coach* creates a new version of the current take with the target words replaced with the audio of the properly emphasized words. Second, if the user did not emphasize the target word in another take, our system applies a parameter-based resynthesis method. This method uses PSOLA [16, 93], a speech processing technique that modifies the pitch and duration of speech by

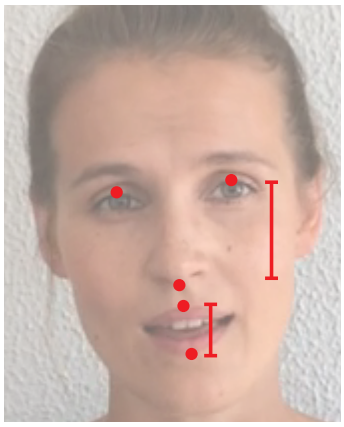

Figure 3.5: *Narration Coach* determines how much the speaker is mumbling by analyzing variance in the speaker's mouth size. The system uses the speaker's face size to normalize the mouth movement measurements.

Figure 3.6: Our system adds emphasis to a word by modifying its vocal parameter contours (red: before; green: after) and creating resynthesized audio using PSOLA.

manipulating small pitch-aligned overlapping windows of the audio signal— moving them closer together or further apart to adjust pitch, and repeating or eliminating windows to adjust duration. Our system constructs new pitch, volume, and duration contours to match typical emphasis contours:

Pitch. Every word contains a stressed vowel phoneme. Our system creates the new pitch contour by setting a pitch peak at the start of that phoneme. The new pitch peak is the maximum pitch of the word plus a parameter p octaves. We construct the contour according to the TILT [33] pitch contour generation model. In this model, the contour follows a piecewise quadratic function ascending from the word's original starting pitch to the new maximum pitch, followed by another piecewise quadratic function descending to the word's original ending pitch (Figure 3.6, left).

Volume. Our system manipulates the volume of the target word by setting a volume multiplier of v from the start of the stressed vowel phoneme through the end of the stressed vowel phoneme. It sets the volume multiplier to 1.0 elsewhere and adds linear transitions to and from the stressed vowel phoneme, so that the volume variations are not perceived as too abrupt (Figure 3.6, center).

Duration. Speakers can extend the typical duration of a word to give it more weight. Our system sets the new duration contour akin to the volume contour, substituting a duration multiplier d for v (Figure 3.6, right). This extends and adds emphasis to the vowel phonemes, which sounds more natural than extending all of the word's phonemes.

Our system parameterizes the contours by the values p, v, and d. We set these defaults to .25 octaves, 1.25, and 1.1 respectively, which represents an audible but not extreme increase in these parameters. If the user wants more control, she can modify the defaults for p, v, and d and fine-tune the synthesized emphasis for each word.

The first phase of emphasis resynthesis has the advantage of using emphasized words that the user said in same context as the target words, and avoids adding artifacts that the parameter-based resynthesis method may introduce; however, it requires that she emphasized the target word in another take and it can introduce artifacts if the speaker had different tones between the takes.

We can apply similar techniques to remove emphasis from words that the speaker over-emphasizes. However, our system focuses on fixing non-emphasized words because novice speakers tend to forget to emphasize words.

Narration Coach modifies the flow of a sentence by creating a faster or slower version of the sentence as needed. It lengthens or shortens the durations of words using PSOLA time-stretching. *Narration Coach* also removes unneeded and awkward pauses in the take, i.e., pauses over a quarter of a second long that do not occur directly after commas or other punctuation.

3.3.5 Constructing the final narration

Narration Coach creates a final narration by playing the best take of each line in succession. Our system defines the best line as the line that follows the most guidelines. A take follows the emphasis guideline if the speaker emphasized each target word. A take follows the variety, flow, and diction guidelines if the respective computed feedback values fall within the "good" range according to the text feedback mappings. If multiple takes follow the same number of guidelines, our system favors the most recent take, and if the user clicks the "star" to mark a take of a line as a favorite, our system uses that take instead. *Narration Coach* mixes room tone into the final narration to prevent audible, unnatural dips to silence in the transitions between sentences.

3.4 Results

We conducted a pilot study where we introduced *Narration Coach* to five users (3 female, 2 male), none of whom has had formal voiceover or audio recording/editing training. We began each session by providing the user with a script and asking her to read it out loud to learn its words and phrasing. We then had her record a narration of that script using Adobe Audition, a traditional audio recording and editing tool. We allowed the user to record additional takes—of the entire script or of specific lines—until she was satisfied with the narration, and we helped her edit the takes together so she did not have to learn how to use the software. After they finished recording the narration with Audition, we loaded the script in *Narration Coach*. We then gave the user a 10-minute demon-

		Recording comparison				*Narration Coach* usage								
Name	Script lines	Takes per line (Aud.)	Takes per line (*NC*)	Final dur. (Aud.)	Final dur. (*NC*)	Start rec.	Open take insp.	Edit dict.	Edit detected emph.	Play take	Play emph. resynth.	Play flow resynth.	Favorite take	Play full narr.
ballotselfies	8	1.375	2	0:56	0:54	10	14	9	2	17	4	3	8	1
blue	8	1	2.125	0:42	0:48	7	32	10	1	26	14	10	8	2
coloradosv	8	2	3.125	0:47	0:57	14	25	4	0	14	2	0	9	1
mosquitos	10	1.8	1.6	0:46	0:55	17	38	14	1	15	2	11	6	3
voting	11	1.36	2	0:58	0:59	12	26	0	1	13	1	3	4	4

Table 3.1: Five participants each recorded two narrations for a script—one using a traditional DAW (Adobe Audition) and one using *Narration Coach*. Users recorded more takes per line and produced slower-paced narrations using our system. We instrumented *Narration Coach* to record usage statistics. "Start rec." is the number of times the user initiated a recording session, and "open take insp." is the count of times the user clicked a line in the script to view the take inspector. "Edit dict." refers to the number of times the user corrected the speech-to-text dictation, and "edit detected emph." tallies times when the user disagreed with and changed the emphasis detection. "Play take," "play emph. resynth.," and "play flow resynth." refer to the user playing different versions of the take within the take inspector. "Star take" tracks the user selecting a favorite take, while "play full narr." tracks when the user listened to the entire narration.

stration of our system's recording, feedback, and resynthesis tools, and asked her to use *Narration Coach* to create a narration. At the end of each session, we solicited the user's feedback on our system's features, and asked her to select whether she preferred her first narration or her narration created in *Narration Coach*. In total, each pilot study session lasted one hour and we compensated each participant with a $15 gift card.

We had the participants record their narrations in Audition before using our system to prevent them from transferring specific feedback from *Narration Coach* to Audition. However, a different type of learning effect is possible in our non-counterbalanced pilot: users may get better at the narration task over time, which could lead to users always producing better narrations in the second task. We tried to avoid this bias by having our users read the script out loud before recording, and by allowing them to re-record the script in Audition as many times as they wanted in order to construct a narration that they liked.

Overall, the users were enthusiastic about our narration tool. Each of the five users preferred the narration they created with *Narration Coach* to the narration they created with Audition, and every user noted that they would use our tool to record narrations. Table 3.1 summarizes the participants' usage of Audition and *Narration Coach* in the recording session. The supplemental material[1] for this research includes the final audio from these sessions. Users demonstrated different usage patterns in *Narration Coach*; for example, some perfected one line at a time while others listened to all available resyntheses for the entire script. They spent more time recording narrations using our system than with Audition, noting that this was because they tried to improve on their takes based on *Narration Coach*'s feedback. They recorded more takes per line in our system than when using Audition.

The recording tools in *Narration Coach* excited the users. Notably, they liked how it organized audio by script line into different takes, how they could start recording from anywhere in the script,

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

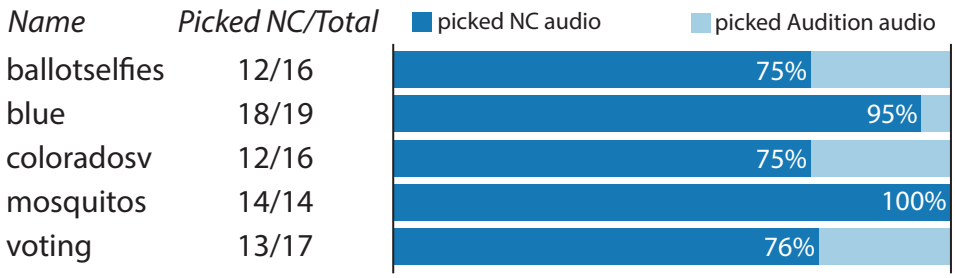

Figure 3.7: Listeners on Mechanical Turk thought the narrations made with *Narration Coach* were higher quality than the narrations made without it for all five of the audio pairs created by participants in our pilot study.

and how they could star a favorite take to play in the final narration instead of manually splicing in the best take as in a timeline-based editor. One participant noted that *"Organizing audio line by line into takes was super helpful because it didn't invalidate an entire recording session if I stumbled over one word."* Another appreciated the "star" feature, as it enabled *"[re-recording] several times without the hassle of splicing the new takes in."* Our system functions well even if the speech transcription has minor errors (e.g., with one or two errors in a take, *Narration Coach* can still compute accurate alignment metadata and thus provide take feedback and resyntheses). However, the automatic speech-to-text transcription was highly inaccurate for one user who had an accent. That user had to manually correct each automatically generated transcript. We expect that off-the-shelf dictation tools will improve as speech recognition research progresses.

Users found the feedback tools useful, as well, with the feedback on speed and awkward pauses being the most useful. One user described that speed detection was useful because *"I usually speak very fast without realizing it,"* and added that emphasis detection was useful because it *"allowed me to make a mental note of whether or not I emphasized certain words."* The participants referenced the pitch variety and diction feedback less often, in part because users did not trust the feedback: *"I... didn't feel like I was very monotone."* Even if the user *was* being monotone or mumbling, our system needs to better communicate feedback that users may find incorrect or even insulting. *Narration Coach* addresses this problem with emphasis detection by allowing users to toggle the feedback if they disagree with the analysis.

Our participants repeatedly used and complimented the automatic flow resynthesis feature that altered a take's speed and removed awkward pauses. However, they found the automatic emphasis resynthesis tool to be less useful. Instead of using the tool as a way to hear what proper emphasis might sound like to guide their future takes, the users treated the emphasis resynthesis as a candidate for a final recording. As such, the users found audio artifacts caused by the parameter-based resynthesis to be off-putting. In a longer recording session, users would likely record more takes of each line, so the emphasis resyntheses would be more likely to use replacement-based emphasis rather than contour-based emphasis, which tends to produce fewer audio artifacts.

Listening evaluation. While all of our participants preferred their narration created with *Narra-*

tion Coach to their narration created with Audition, we sought additional input from crowd workers. For each script, 20 US workers with over 95% approval rates on Amazon Mechanical Turk listened to both versions of the narration. We asked them to select which clip had the higher overall narration quality. We rejected workers that did not spend enough time on the task to listen to both clips in their entirety, which left an average of 16 workers per narration pair. Figure 3.7 shows the proportion of listeners that selected each narration. For each narration pair, we performed a binomial test to see if the proportion of listeners that selected the *Narration Coach* recording was greater than chance (50%). In all five cases, more listeners preferred the *Narration Coach* audio clip to the other clip ($p < .05$).

3.5 Conclusion

High-quality narrations are integral to creating captivating digital content, but novice users are not aware of the guidelines necessary to produce such voiceovers. We have presented *Narration Coach*, an interface to help users create narrations by providing text and audio feedback throughout the recording process. Using our system, novice users can better create high-quality narrations that more closely adhere to professional voiceover guidelines.

Chapter 4

Editing speech

In the second stage of the iterative production pipeline, the producer edits her recorded speech to assemble her story. The recorded speech includes both narration and interview footage. An experienced producer selects and combines the most salient content and then refines individual sentences to improve the phrasing and rhythms of the speech.

4.1 Challenges in editing speech

Traditionally, the producer maintains a log of her footage in order to find specific content. This log contains notes and quotes and their associated filenames and timestamps. The producer creates this log manually, and if she wants to find a particular line in unlogged footage, she needs to scrub through the audio by hand.

Current audio editing tools force the producer to sequence and manipulate the speech at the level of the audio waveform. As a result, the producer must map her 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 tedious and time-consuming.

As she edits speech, the producer also tries to maintain natural patterns of speech, i.e., speech with appropriate pauses and breaths. For example, without pauses and breaths between sentences, speech sounds rushed and robotic. The producer inserts and edits these breaths and pauses where necessary, again using low-level waveform operations.

We have developed a speech editing interface to help the producer edit raw speech tracks into audio stories. Our interface analyzes the content of the speech in order to allow the producer to work at a much higher level. To help the producer navigate and edit raw speech recordings, our interface includes a transcript view of each speech track. As in previous transcript-based speech editing systems [11, 22, 98], the producer can directly modify the transcript text, and our system propagates the corresponding edits to the speech waveform. However, unlike previous systems, our interface 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

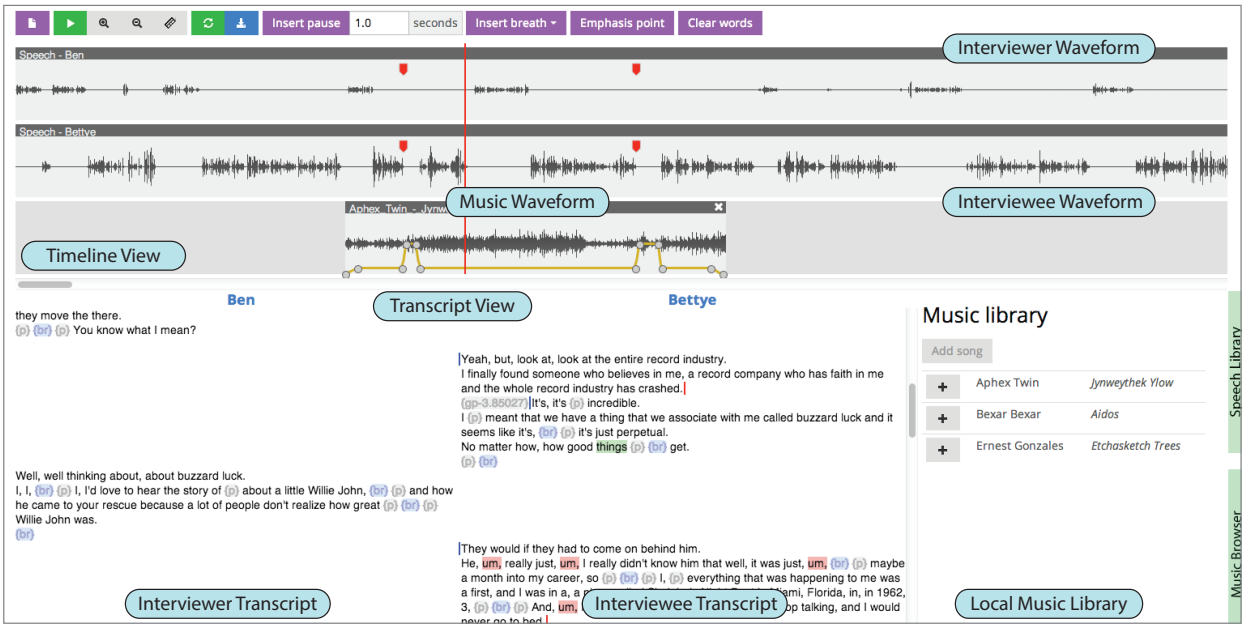

Figure 4.1: Our editing interface features two views of each speech track: a traditional waveform view, and a text-based transcript view.

the speech to help the producer refine phrasings while maintaining a natural sounding rhythm. We first describe our speech editing interface, and then present the content-based analysis algorithms that enable the interface's tools.

4.2 Transcript-based speech editing interface

The producer starts by loading raw speech tracks into the editing interface. In Figure 4.1, 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 4.2). 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 cor-

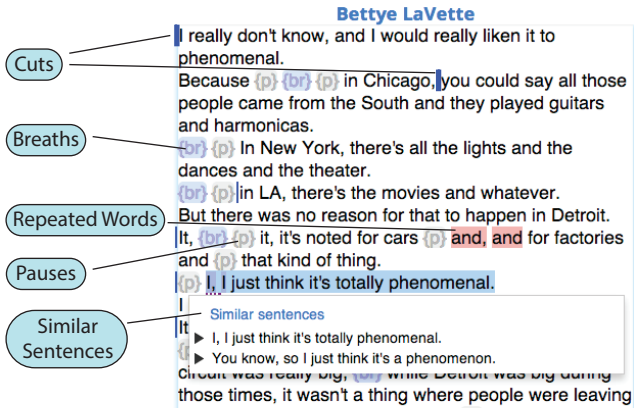

Figure 4.2: The transcript view allows the producer to edit the story at the word level. This view marks cuts, breaths, pauses, repeated and unnecessary words, and similar sentences, all of which enable the producer to quickly edit the speech.

responding 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. A producer usually captures room tone at the start of a recording session and our system automatically treats near-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. The producer 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 4.2). 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 4.2).

The transcript view also explicitly marks the breaths and pauses that occur in each speech track (Figure 4.2). 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 [4]. 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 period (' . ') 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 the speaker just said. Thus, the producer may choose to insert pauses at different points in the speech, to emphasize particular thoughts 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.

4.3 Algorithmic methods for speech editing

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

4.3.1 Obtaining the transcript

Our editor requires a text transcript for all of the speech footage. Speech footage recorded in *Narration Coach* already has an associated transcript. However, other narration footage and interview footage does not have this metadata. We obtain a transcript of the raw speech tracks using the crowdsourced transcription service, CastingWords.com [23]. 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, we designed our audio editor to highlight such words in the transcript and let the producer 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.

4.3.2 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) [104], a tool built using HTK (HMM toolkit), a speech recognition library [103]. 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 [79] 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.

4.3.3 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.

4.3.4 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. While *Narration Coach* groups these multiple takes while the speaker records a narration, our editing interface finds these repeated takes in all forms of speech footage. 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.

4.3.5 Rendering edited speech audio

When rendering edited speech, 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.

4.4 Informal evaluation of the speech editor

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 encouraging. Each participant was able to successfully produce a high-quality audio story that contained significant edits to the speech. 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 felt that the speech editor 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)."*

In addition to performing this informal evaluation, we used our tool to create fully edited audio stories. See Section 5.8.2 for details.

4.5 Limitations

We found that the main limitation in our editing tool is that errors in transcription or in alignment sometimes lead to our system generating audible artifacts (e.g., clicks, noise, etc.) in the edited speech. The producer can solve these problems by correcting transcription errors in *Narration Coach* before using our speech editor. We could also provide an interface that allows users to correct inaccuracies in the speech/text alignment.

4.6 Conclusion

Our speech editing interface helps the producer to create high-quality audio stories. A key aspect of this interface is that it analyzes the content of the raw speech to provide higher-level editing capabilities than a general-purpose DAW like Avid Pro Tools. With our editor, the producer 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 waveform.

Chapter 5

Editing music

In the final stage of the iterative audio production pipeline, the producer adds and edits music. Professional producers add music to audio stories to add ambience, emphasize important moments in the speech, and heighten the emotions of the speech.

5.1 Challenges in music editing

The producer may add a music track to provide ambience for an audio story. After the producer finds a segment of music that provides her ideal ambience for part of the story, she may find that segment is too short to score the target speech segment. In this case, she can either try to find a longer music track that provides the desired ambience, or she can extend the original music segment by looping it. To manually loop a music segment, the producer listens carefully to find two initial cut points for the beginning and end of the loop. Then, while listening to the loop, she finely adjusts the cut point forward or backwards until the looping sounds seamless. This time-consuming manual operation requires sample-level audio manipulations.

In order to emphasize an important moment in the speech, the producer may use a technique called a musical underlay (Figure 5.1). In a musical underlay, the music track contains three segments: (1) a music pre-solo that fades in before the emphasis point, (2) a music solo that starts at the emphasis point and plays at full volume while the speech is paused, and (3) a music post-solo that fades down as the speech resumes. At the beginning of the solo, the music often changes in some significant way (e.g. a melody enters, the tempo quickens, etc.). Aligning this change point in the music with a pause in speech and a rapid increase in the music volume further draws attention to the emphasis point in the story. To create an effective musical underlay, the producer must take the time to listen to the desired music track multiple times to find the best music change points, and then fine-tune the music's position and dynamics with respect to the speech.

While the two above techniques involve editing and adding music for local sections of the music, an overarching goal of a musical score in audio stories is often to globally highlight and heighten the emotions of the speech. To achieve this, the producer crafts her musical score so the emotions in the music match the emotions in the speech. She also ensures that the music meets other design

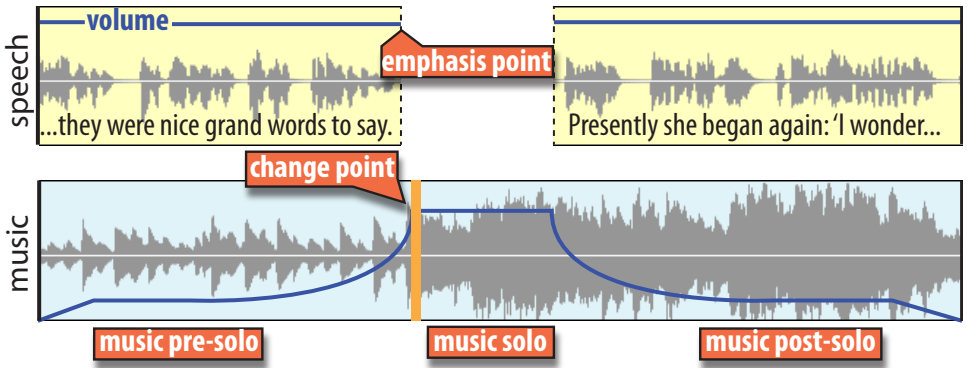

Figure 5.1: An example of a musical underlay.

principles of effective musical scores, e.g., adding musical underlays and not playing music for too long at once. This process for creating a complete musical score for an audio story is time-consuming and challenging for expert producers, while novice producers are unaware of the intricacies and design guidelines that comprise a well-constructed score.

5.2 Overview of structure-based music retargeting

We have built tools that address each of these methods for integrating music into audio stories. These tools rely on re-arranging ("retargeting") the music's beats to create a new version of the music track that better fits with the speech. To perform retargeting, we translate target constraints into numeric costs. The two primary costs are *matching costs*—costs of how well the music aligns with the speech—and *transition costs*—costs for beat-to-beat transitions in the music

We specify our optimization and constraints on beats, which are short, structural units of time in music. Formally, we consider a piece of music as a set of n beats $\mathbb{B} = \{b_1, b_2, \ldots, b_n\}$ where $b_1, b_2, \ldots, b_n$ is the natural order of beats in the music. To detect beats in a piece of music we apply Ellis's beat tracking algorithm [35]. This algorithm first estimates the music tempo. Then, it applies dynamic programming to find optimal beat times that roughly maintain the estimated tempo. Beats lengths within a song vary because tempo often shifts within a music track, so we use the average beat length as our unit of time throughout our music retargeting algorithms.

We store the *transition costs* in table T, with size $n \times n$ (Figure 5.2). Entry $T_{i,j}$ is the cost for moving from beat b_i to beat b_j and is independent of the location in the output score. The transition cost encodes whether the music could make a seamless-sounding jump from b_i to b_j.

A transition from beat b_i to another beat b_j sounds natural when the timbre, pitch, and volume of b_j are similar to the timbre, pitch, and volume of b_{i+1}, the next beat in the original beat sequence. We compute the transition cost table T by combining timbre, pitch, and volume distances (D_t, D_p, and D_v) for each pair of beats:

$$T_{i,j} = w_t D_t(i+1, j) + w_p D_p(i+1, j) + w_v D_v(i+1, j).$$

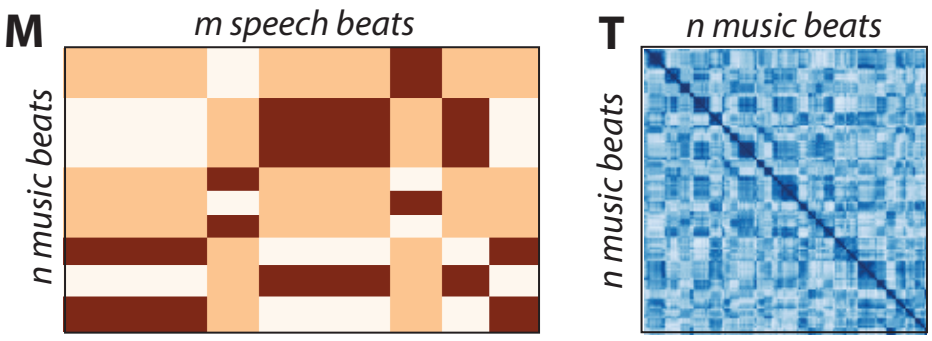

Figure 5.2: The matching cost table M (left) has lower cost when music beat b_i and speech beat k have the same emotion, or when b_i is a music change point and k is a speech emphasis point. The transition cost table T (right) gives the cost of moving between beat b_i and beat b_j. These examples show transition and matching cost tables with only the transition and matching constraints. Darker colors imply cheaper costs.

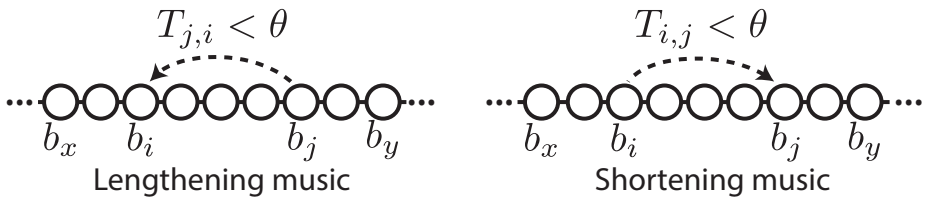

Figure 5.3: For simple music retargeting, we lengthen music by finding low cost transitions to earlier beats in the music. The producer can click a button to repeat such loops in the track. To shorten music, we find low cost transitions to later beats. The producer can then choose to delete the in-between beats.

Our algorithm estimates timbre differences by computing MFCCs [61] for each beat, and then computing the cosine distance $D_t(i, j)$ for all pairs of beats b_i and b_j. We similarly estimate pitch differences by computing chroma features [35] for each beat, and then computing the pairwise cosine distances $D_p(i, j)$. Additionally, a large volume difference between beats can create a jarring listening experience. We compute the RMS energy, a measure of volume, of each beat and then take the absolute energy difference between all beat pairs (b_i, b_j) to get $D_v(i, j)$. These transition cost constraints ensure that the original beat progression—a natural sounding progression—has zero transition cost (i.e., $T_{i,i+1} = 0$ for all i).

5.3 Simple music retargeting (looping)

In simple music retargeting, the producer wants to extend a section of music. We enable this by automatically finding seamless loops within that section. As shown in Figure 5.3, given a music segment from beat b_x to b_y, we look for beats b_i and b_j where $x \leq i < j \leq y$ and the transition cost $T_{j,i}$ is low. This ensures that playing the music from beat b_i to beat b_j and then looping back

to b_i sounds natural. Likewise, if the producer wants to shorten a section of music, we can look for lost-cost forward transitions in the segment (Figure 5.3, right).

If b_j is a small number of beats after beat b_i, the resulting loop may sound unnaturally repetitive. To avoid creating such loops, we can restrict our search to beats where b_i and b_j are at least, for example, 8 beats apart.

5.4 Constrained music retargeting (musical underlays)

The simplest form of constrained music retargeting is a *musical underlay*. A musical underlay has two main parameters: the speech emphasis point and the music change point.

5.4.1 Composition of a musical underlay

We draw on publications describing the best practices of radio production [2, 4, 5, 12, 44] and analyze high-quality radio programs [3, 6, 43] in order to extract guidelines for each step of the underlay creation process.

Marking speech emphasis points

Speech emphasis points occur after important moments in speech that present the central ideas, introduce new characters, or set the mood. The producer often emphasizes these points with an underlay. A short break in the speech allows listeners to process the content of the story and can separate long passages into shorter chunks that are easier to understand [4].

Selecting music

Music serves multiple functions in an underlay. It augments the emotional content of the story and often builds tension leading up to the emphasis point in the speech [5]. The music usually exhibits a significant change at the emphasis point to further draw the listener's attention. This change point can correspond to a melody entering, the tempo quickening, new instrumentation beginning, or a strong downbeat [4].

Given a speech emphasis point and a music change point, the producer creates an underlay by aligning the audio tracks and setting their dynamics.

Aligning music and speech

Aligning music to speech requires choosing the entry and exit points of the music, positioning the change point in the music with respect to the emphasis point in the speech and determining the length of the music solo. The entry point of the music pre-solo often corresponds with a rising action or change of tone in the speech. Abel and Glass [4] suggest that the pre-solo should start 12 seconds before the emphasis point in the speech. The change point in the music usually appears slightly after the emphasis point in the speech. This gap ensures that the change in music does not

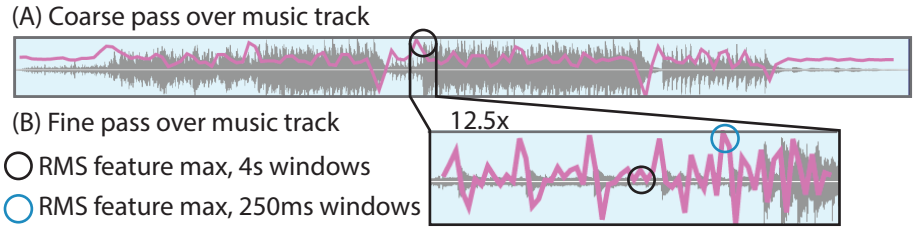

Figure 5.4: To (A) generate rough music change point estimations, our system finds the maxima of the RMS energy feature using 4-second subwindows that overlap by 50% and span the entire music track. To (B) refine a change point, our system applies the same approach but with 250 ms subwindows that span an 8 second window about the coarse point. This two-step process first finds large scale changes in volume and then find the strongest downbeat at the local scale.

interfere with the speech and can add to the dramatic effect of the change in music. Finally, Abel and Glass suggest that the music solo should last about 6 seconds before the speech resumes. The length of the music post-solo is less consistent. In some cases the end of the music signals another emphasis point in the speech.

Adjusting dynamics

The goal of adjusting the underlay dynamics is to further heighten the impact of the change point in the music while ensuring that the speech is audible. During the pre-solo the music plays softly, gradually increasing in volume. Near the emphasis point the volume quickly increases and reaches its maximum at the change point in the music to create a strong auditory transition. Finally, just before the speech resumes the music quickly fades down for the post-solo.

5.4.2 Automatic underlay creation

We designed a music retargeting algorithm to automate parts of the musical underlay creation process. The producer provides a speech emphasis point and a music track. We then automatically find a music change point in that track, align the music and speech, and adjust the dynamics.

We investigate two methods for finding music change points. The first method involves a two-phased coarse-to-fine pass over the music track [76]. We break the music track into 4 second windows with 50% overlap. We then compute RMS energy, MFCCs, and chroma features on each window, and compute the distance between adjacent windows for all three features (signed difference for RMS energy, and Euclidean distance for both MFCCs and chroma). For each feature, we identify the consecutive pair of windows yielding the maximum distance, which gives us an 8-second long combined window with a large musical change (Figure 5.4A). Then, to find the exact change point, we repeat the same procedure on the 8-second segment with 250 ms windows instead of 4-second windows. We set the music change point to the start of the second window in the consecutive pair with the largest distance (Figure 5.4B). We can find multiple music change points for each feature

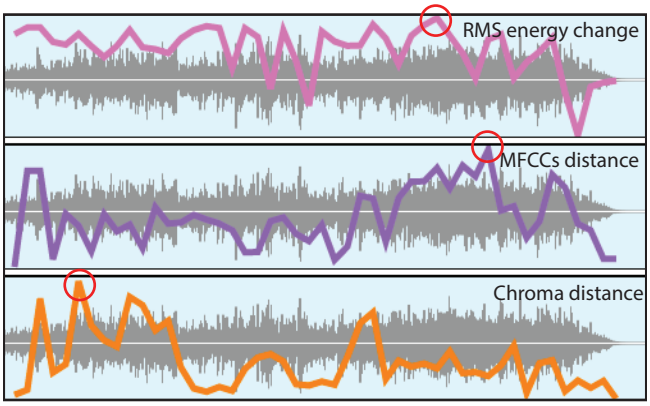

Figure 5.5: In the music track "Scrapping and Yelling" by Mark Mothersbaugh, the maximum of the RMS energy change feature does not correspond to a strongly perceptible change in music. However, the maxima of both MFCCs and chroma distance features give a strong sense of change and lead to an effective underlay.

by finding the top n distances instead of just the top distance in the first part of the procedure. This algorithm gives us potential music change points for each of the three types of features.

We have found that RMS energy often works well for finding music change points because it identifies large changes in volume at the coarse level and strong downbeats at the fine level (Figure 5.5). We use it as our default feature, but we have found that MFCCs and chroma are useful for music whose variation is primarily in timbre or harmonics rather than in volume. We also investigated Foote's algorithm [40] for finding "novel" points in a music track. This algorithm is similar to our two-pass algorithm, but incorporates coarse- and fine-grained searching into a single pass.

Once we find a music change point, we align the music so the change point plays 500 ms after the speech emphasis point, where we add a 6 second pause in the speech. We adjust the dynamics so the music slowly fades in, steeply crescendos to full volume at the music change point, steeply decrescendos to a soft volume after the music solo, and finally fades back out. By default, we set the lengths of these fades to match producer guidelines (12-second fade-in, 6-second solo, 12-second fade-out) but these parameters are user-customizable.

5.4.3 Underlays for multiple speech emphasis points

Our technique for creating musical underlays emphasizes one point in the speech with a point in the music. However, in some cases, a producer may hope to use a single, continuously playing piece of music to emphasize multiple speech emphasis points. The original progression of the music likely does not feature two music change points that occur with exactly the duration difference of the two speech emphasis points.

In order to retarget music to fit these constraints, we extend the retargeting approach we set up earlier for simple, loop-based retargeting. We introduce the matching cost table M with size $n \times m$. Here, m is the number of music beats required to score the speech. We set m to the length of the speech we want to score divided by the average beat length of the selected music track. We call each

of these m intervals in the speech a "speech beat." Entry $M_{i,k}$ is the cost for playing beat b_i at speech beat k in the output score. Most of these costs are a constant—we have no particular preference for playing a music beat b_i at any speech beat k in the story. If b_i is a music change point and speech beat k is a speech emphasis point, we increase $M_{j,k}$ by a constant for all $j \neq i$.

Our algorithm searches for a sequence $\boldsymbol{s}$ of m beats that minimizes the matching and transition costs; it then generates the output musical score by re-sequencing the original beats according to $\boldsymbol{s}$.

More specifically, we search for the sequence of beat indices $\boldsymbol{s} = [s_1, \ldots, s_m \,|\, s_i \in \{1, \ldots, n\}]$ of m beats that minimizes

$$cost(\boldsymbol{s}) = \sum_{k=1}^{m} M_{s_k,k} + \sum_{k=1}^{m-1} T_{s_k,s_{k+1}} \tag{5.1}$$

where k is the speech beat index in the output score. A recursive form of this equation enables us to apply dynamic programming to find the optimal beat sequence. So, the recursive minimum cost of a length-m sequence $\boldsymbol{s}$ ending with beat b_j is

$$c(j, m) = \min_{i \in \{1,\ldots,n\}} \left(c(i, m-1) + M_{j,m} + T_{i,j} \right).$$

The right-hand side is a minimum over three terms. The first term, $c(i, m-1)$ is the minimum cost $(m-1)$-length sequence that ends in beat b_i. The second and third terms, $M_{j,m} + T_{i,j}$ are the matching and transition costs for the last beat b_j. The optimal sequence $\boldsymbol{s}$ with any ending beat then has cost

$$\min_{j \in \{1,\ldots,n\}} c(j, m).$$

Because we have expressed the minimum cost sequence recursively, in terms of its subproblems, we can apply dynamic programming to find the optimal beat sequence.

So far, we encode constraints on natural music transitions in T and constraints for playing music change points at speech emphasis points in M. Therefore, the optimal cost output beat sequence is a musical score with natural music transitions, and tends to align music change points with speech emphasis points. As desired, these musical scores use a single music track to highlight multiple speech emphasis points with music change points. We adjust the dynamics around each music change point as we did with a single underlay.

5.5 Emotionally relevant musical scores

Loops and musical underlays are both techniques for matching music to a small section of speech. However, in highly produced audio stories, the entire musical score emphasizes and adds nuance to the emotions of the speech. Crafting a full, emotionally relevant musical score involves smoothly re-sequencing, looping, and timing the music to match the emotions in the story as they change over the course of the narrative. This process requires significant audio production expertise and is challenging even for expert producers. As a result, most of the recorded stories available today focus on providing the speech track and do not contain a musical score.

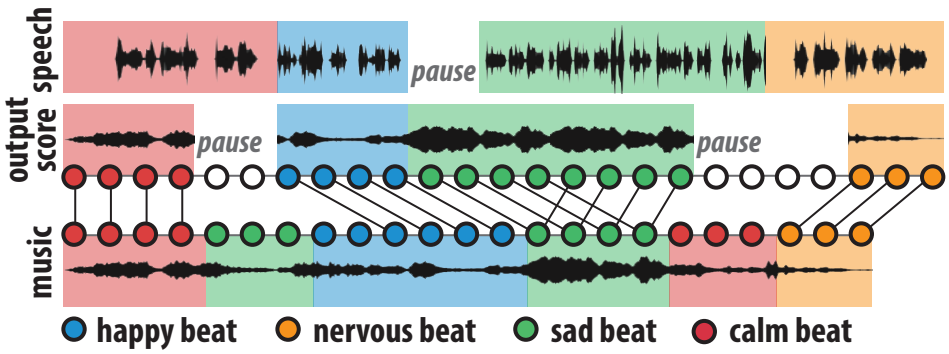

Figure 5.6: Our algorithm re-sequences the beats (circles) of the input music (bottom row) to match the emotions of the speech (top row). Our algorithm inserts pauses in speech and music, and makes musical transitions that were not in the original music in order to meet these constraints.

We developed a system to allow users with no audio editing expertise to generate high-quality, emotionally relevant music scores for audio stories. To generate a score, our system requires a speech track with associated emotion labels, and one or more music tracks with associated emotion labels. Our system extends our music retargeting dynamic programming algorithm to re-sequence input music tracks so that the emotions of the output score match those of the speech (Figure 5.6). We studied how expert producers use music in audio stories and incorporate additional constraints into the optimization that correspond to structural and stylistic design decisions in their scores.

We demonstrate the effectiveness of our score generation tools by generating a total of 20 musical scores including examples using each of our three labeling methods. We ask listeners to rank the overall quality of these results. Listeners rank the results generated with our crowd-labeling technique as high or higher than the hand-labeled results, which are in turn better than results using our automatic approach and no music. Our system generates scores that are preferable to no musical score for all of the speech tracks. We also investigate the inter-labeler reliability of agreement and find that human-produced labelings tend to agree more with each other than with our automatic labelings.

5.5.1 Labeling emotions

Figure 5.7 shows our pipeline for generating emotionally relevant musical scores. A user begins by selecting a speech track such as an audiobook, and one or more instrumental music tracks (i.e., music without lyrics). Our system then gathers emotion labels for each of the speech and music tracks. After collecting these labels, our system constructs a musical score by re-sequencing the input music tracks so the emotion labels of the output music match the emotion labels of the speech.

Our system offers three methods for obtaining the required emotion labels: hand-labeling, crowd-labeling, and automatic labeling. These three methods offer trade-offs in time, effort, and personalization. Hand-labeling produces emotions that reflect the user's emotions and may result in the most personal musical score, but it takes the most time for the user. The crowd-labeling

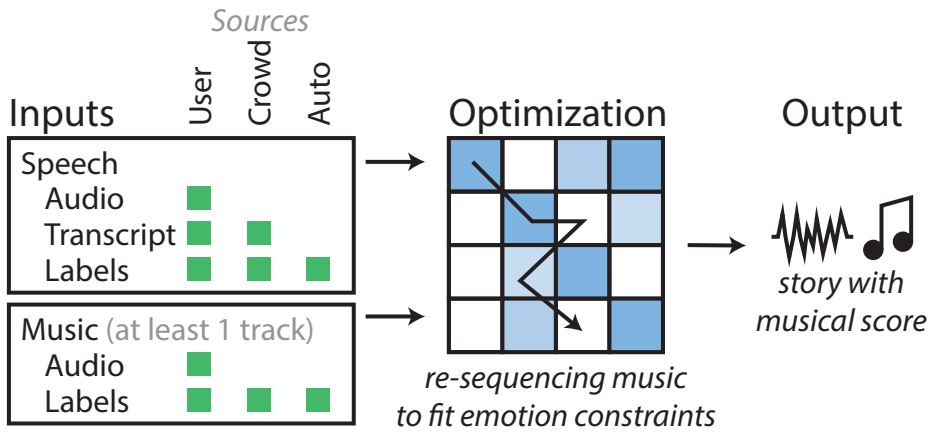

Figure 5.7: Our automated musical score generation system requires a speech track and music tracks, and emotion labels on those tracks. Our algorithm re-sequences the music tracks to create an output musical score that matches the emotions of the speech. The green boxes denote the sources—user, crowd, and fully automatic—that our system provides for obtaining each input. Our system also requires a speech transcript so users and crowd workers can more quickly label the emotions by reading the text instead of listening to the speech.

method requires no extra work on the user's part and incorporates human ratings. However, this method takes more time than hand-labeling to acquire emotion labels. Finally, the fully automatic method produces labels immediately, but they may not accurately reflect the personal emotions that the user—or any human—feels about the story and the music. All of these methods require a speech transcript, which may cost money to obtain. The crowd-labeling method has the added cost of paying workers.

Affect researchers have found Russell's circumplex model [80] of quantifying emotion with 'valence' and 'arousal' to be highly consistent with behavioral and cognitive neuroscience study results [70]. The valence dimension indicates whether an emotion is positive or negative, whereas arousal indicates the intensity of the emotion. In our systems, users do not need to learn the circumplex model. Instead, we focus on four emotions, *happy, nervous, sad,* and *calm*, because they almost evenly span the circumplex [80]. As shown in Figure 5.8, we set their coordinates to $(.95, .31)$, $(-.31, .95)$, $(-.95, -.31)$, and $(-.95, .31)$, respectively, an equal distribution near their original locations [70]. The circumplex and other emotion models likely do not generalize across cultures. Even within a culture, people may disagree on emotion definitions. Despite these possible differences, we use the four contrasting emotions as a starting point to explore emotion-based constraints for generating scores.

Speech emotion labels

Our goal is to break the speech into segments of similar emotions. Segmenting a speech track into emotionally similar segments based solely on the raw waveform requires careful listening and can be

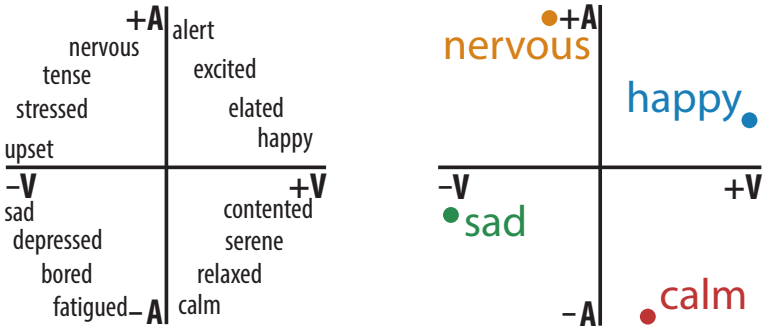

Figure 5.8: The valence/arousal circumplex (left) parameterizes emotions into two dimensions: valence, a measure of positivity, and arousal, a measure of intensity. In our system, we used four emotions that were nearly evenly spaced on the circumplex (right).

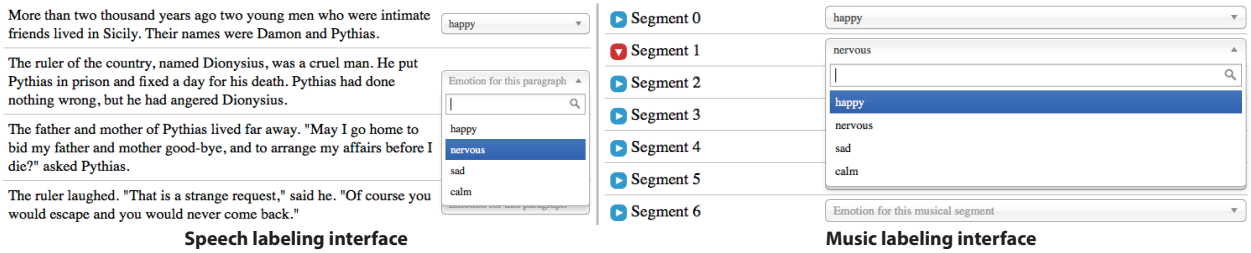

Figure 5.9: Our interface for labeling speech emotion (left) asks the labeler to annotate each paragraph with an emotion. Our interface for labeling music emotion (right) asks the labeler to label each music segment with an emotion.

difficult. In text, however, paragraphs usually represent topically coherent segments that convey one predominant emotion. Therefore, we use the paragraph boundaries in the speech's text transcript to segment the speech. The user chooses one of the three labeling methods to get emotion labels for each speech segment.

Hand-labeling the speech. If users wish to personalize the emotion labels of the speech, they can label the emotion of each paragraph of the text transcript by hand (Figure 5.9).

Crowd-labeling the speech. To crowd-label the emotions of the speech, our system posts tasks to Amazon's Mechanical Turk. These tasks are identical to the hand-labeling interface (Figure 5.9).

Crowd workers often try to complete tasks as quickly as possible so they can maximize their wages. Such workers may not fully engage with the speech or its emotions and produce inaccurate emotion labels. Yet, emotions labels can be subjective, and there is no way to test the accuracy of a worker's labeling. Our approach is to obtain emotion labels from multiple workers and then select the labeling of a single worker that best represents all of the workers.

We interpret each worker's complete speech labeling as a sample from the distribution of possible speech labelings, first assigning each paragraph i a distribution p_i where $p_i(l)$ is the probability that

a worker labeled paragraph i with emotion l in all of the labelings we have collected. For example, if we have ten workers total and 7 of them label paragraph 1 as 'sad' and 3 label it as 'calm'—then $p_1(sad) = 7/10$, $p_1(calm) = 3/10$, and $p_1(happy)$ and $p_1(nervous)$ are both zero. We then find the worker that gave the most probable labeling according to this distribution. Suppose $\ell = \{l_1, \ldots, l_k\}$ is a worker's labeling of the sequence of paragraphs in the speech. We compute the probability of this labeling as

$$P(\ell) = \prod_{i=1}^{k} p_i(l_i).$$

Our system computes this probability for each worker's labeling and assigns the labeling with the highest probability to the speech. An alternative to our *most probable worker labeling* approach is to choose the most frequently labeled emotion for each paragraph independently. However, such a voting scheme can yield emotion transitions between paragraphs that did not appear in any of the individual worker's labelings. In contrast, our approach yields a labeling guaranteed to be consistent with at least one worker's labeling. This strategy of picking one representative rather than averaging or aggregating worker results appears in other recent work to ensure self-consistent results [58, 100].

Automatically labeling the speech. To automatically label the speech, our system estimates the emotion of each paragraph based on the emotion conveyed by each word. Warriner et al. [94] have collected a corpus of crowdsourced valence and arousal ratings for nearly 14,000 English words. We normalize these scores by the global valence/arousal mean and standard deviation. We then compute the average of the normalized valence/arousal scores of all words in a paragraph. Our system projects those averages to the nearest of our four labels to obtain an emotion label for each paragraph.

Music emotion labels

Our system provides techniques for music emotion labeling similar to our speech labeling methods. As in speech labeling, we first break the music into segments. Structural segments of music often contain one predominant emotion because the features that differentiate structure—timbre, pitch, volume, and self-similarity—are also indicative of music emotion. Following McFee and Ellis [64], our system segments music by computing a hierarchical clustering of self-similarities in a track and finding an optimal pruning of the cluster tree. Then, the user selects one of the three labeling methods to get emotion labels for each music segment.

Hand-labeling the music. If users wish to personalize the emotion labels of the speech, they can listen to and assign labels for each music segment (Figure 5.9). As in hand-labeling the speech, this method is preferable for users that have time and desire a personalized musical score.

Crowd-labeling the music. To crowd-label the emotions of the music, our system asks workers on Mechanical Turk to listen to and label the emotions of the music segments for an entire track (Figure 5.9). Our system then selects a final emotion labeling by finding the worker's labeling that best represents all of the worker labelings (see earlier section on *Crowd-labeling the speech*).

Automatically labeling the music. Schmidt et al. [82, 83] have developed methods for automatically predicting the valence and arousal of music. Their MoodSwings Turk dataset consists of a large set of crowd-generated, per-second valence/arousal labels and accompanying audio signal

processing features (MFCCs [61], spectral contrast [53], and chroma [35]). Our goal is to automatically predict the emotion of each music segment, but the dataset contains per-second labels and no notion of segments. We follow Schmidt et al. [82] and train a multiple linear regression model on the MoodSwings Turk dataset to predict per-second emotions.

Emotion in music has a time dependency. That is, emotions at times before time t influence the emotion at time t. To account for this dependency, our model uses ten seconds (times $t-9, \ldots, t$) of per-second MFCC features to predict the valence and arousal at time t [82]. Our system predicts the valence/arousal at each second of each music segment and then finds the average of the predictions. Because of a limited corpus of music in the training set, this model often reports that all segments of a track have the same emotion whereas human labelers give more varied labelings. We subtract the overall segment average from each music segment's average to get a new, more varied prediction. Then, we project each prediction to the nearest of our four emotion labels to get a label for each segment.

5.5.2 Generating the musical scores

We designed our score generation algorithm to generate musical scores that match the emotion of the speech while also following other characteristics of high quality musical scores. We identified these characteristics by listening to hours of expertly produced audio stories, noting structural and stylistic patterns in their musical scores. We further studied books and online sources on audio story production to refine these characteristics. Figure 5.6 shows an example of how our algorithm re-sequences input music to match the emotions of the speech. To generate an emotionally relevant score, we extend our music retargeting algorithm with emotion matching and other constraints that modify the transition cost and matching cost tables.

Emotion matching costs

The main purpose of our system is to create music that matches the emotions of the accompanying speech. The speech and music contain emotion tags (happy, nervous, sad, and calm) and their respective numerical values in valence/arousal space. We set matching costs in table M so the valence and arousal of the optimal output score matches the valence and arousal of the speech as closely as possible.

To encode this constraint in our optimization, we compute the ℓ_2 distance $D_{va}(i,k)$ between speech valence/arousal and music valence/arousal at each music beat b_i and speech beat k. We set matching cost $M_{i,k} = w_{va} D_{va}(i,k)$, where w_{va} is a constant that controls the importance of the matching cost in our optimization.

Structural constraints

Our algorithm can generate musical scores that match the emotions of speech using only the basic matching cost and transition cost constraints. However, high-quality scores contain additional structure by using pauses and limiting music segment lengths.

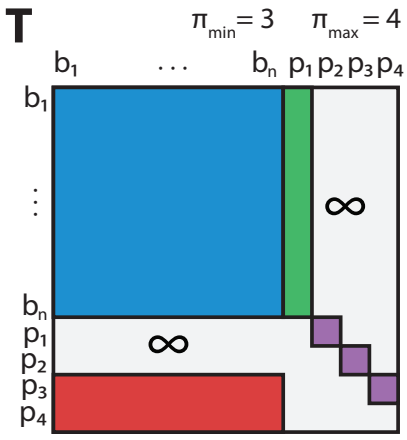

Figure 5.10: Our algorithm enables pauses in the musical score by extending the transition cost table T with pause beats.

Pauses. In most high-quality audio stories, the music is not constantly playing. Instead, music occasionally fades in and out to create sections of the story without music. These pauses call attention to the fact that the musical score exists, forcing listeners to think about the music and its emotions.

Figure 5.10 shows how we adjust our transition cost table T to accommodate pauses in a musical score. We first define π_{min} and π_{max}, the minimum and maximum beat lengths of pauses, respectively. We concatenate π_{max} rows and columns to the original T (the blue block in Figure 5.10 is the original T). The π_{max} new beats are pause beats. Each of these added beats p_i represents the ith beat of a pause.

Every pause in music starts at the first pause beat; any beat in music can transition to beat p_1 but no other pause beats (green rectangle in Figure 5.10). To maintain the invariant that p_i is the ith beat of a pause, we only allow p_i to transition to p_{i+1} (purple squares). Any pause beat at or above the minimum pause length π_{min} has a free cost transition back to any music beat (red rectangle). In practice, we assign a cost κ_{start} to entering a pause (green rectangle) that is greater than the cost of a natural-sounding music transition (see *Setting Parameters*, below). This cost ensures that our score does not add pauses when it can play good-sounding music. We also add a small cost of κ_{intra} to transitions between p_i and p_{i+1} if $i > \pi_{min}$. Without this cost, most pauses have length π_{max} because the intra-pause transitions are free while optimal sequences in music often contain non-free transitions.

Music segments. High-quality audio stories tend not to play overly short or long segments of music. Short segments do not give the music time to integrate with the speech and its emotions, while overly long segments can cause the listener to "tune out," or ignore the music. Our algorithm constrains the length of music segments in the output score.

Figure 5.11 shows how we constrain the length of a music segment to at least δ_{min} beats and at most δ_{max} beats. In order to keep track of the lengths of music segments, we create a new transi-

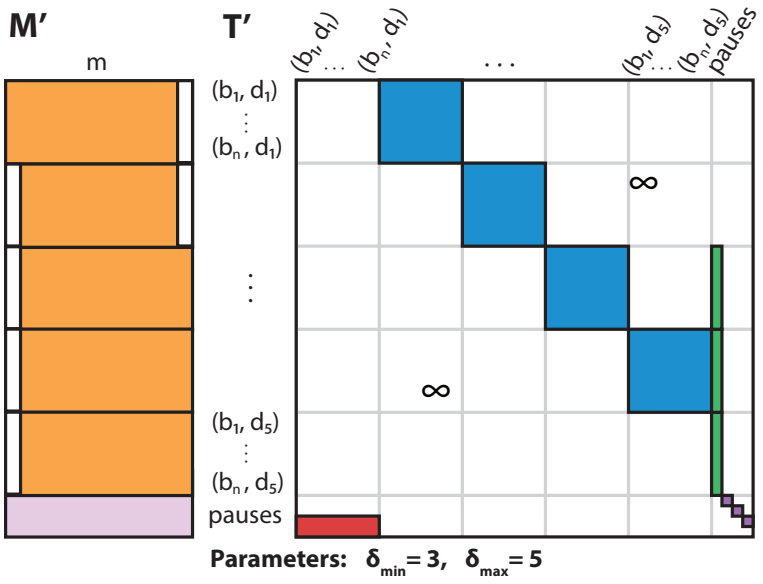

Figure 5.11: We provide constraints on the length of music segments by creating a new matching cost table M' and transition cost table T'. The indices of the new tables are (beat, segment-length) pairs rather than just beats. We build the M' table from δ_{max} copies of M, except that the music must start at beat-length d_1 and end at beat-length greater than δ_{min}. We copy blocks of table T in Figure 5.10, indicated by colors, to create table T'.

tion cost table T'. Each index in the table represents a (beat, segment-length) pair (b_i, d_k) where segment length d_k represents the kth beat in the current music segment. We construct this new table by copying blocks from the existing transition cost table (the colored blocks from Figure 5.10 correspond to the colored blocks in T' in Figure 5.11).

To maintain the invariant that a beat a segment length d_k is the kth beat of the music segment, we only allow (b_i, d_k) to transition to (b_j, d_{k+1}) for any beats b_i and b_j (blue squares in Figure 5.11). Once the music has been playing for δ_{min} beats, a pause can begin. Any (beat, segment-length) pair that is at or above the minimum segment length δ_{min} can transition to the first pause beat p_1 (green rectangles). We copy the intra-pause transitions directly from T (purple squares). Finally, a pause must transition back to a (beat, segment-length) pair at segment length d_1, because all music segments start with length 1 (red rectangle).

We also create a new M' with constraints for music segment lengths. The first (beat, segment-length) pair of the output score must be at music segment length 1. Otherwise, we could not enforce the minimum length on the first music segment in the output score. The last (beat, segment-length) pair of the output score must be at music segment length at least δ_{min}, so the last segment of the score is longer than the minimum segment length. We set the matching costs to infinity where these conditions do not hold (see M' in Figure 5.11).

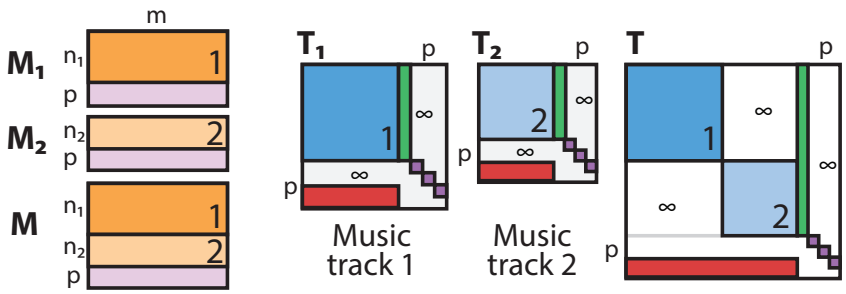

Figure 5.12: Our algorithm generates scores composed of multiple music tracks. It first computes tables M (left) and T (right) for each music track using a consistent speech beat unit, and then combines the tables to form the final M and T. This construction of T does not allow inter-song transitions, but does allow for efficient optimization.

Stylistic constraints

Expertly created scores employ specific stylistic techniques in order to improve the overall score quality. We have designed constraints to imitate these techniques.

Minimum loop constraint. High-quality musical scores often contain loops to allow a short section of music to score a longer section of speech, but if these loops are too short, the music can sound unnaturally repetitive. We introduce a minimum loop constraint to prevent the score from looping sections of music that are less than eight beats long by setting $T_{i,j} = \infty$ if $i - 8 < j \leq i$.

Musical underlays. While we discussed a method for adding musical underlays locally, we also enable musical underlays in our global score optimization. We first preprocess the input speech, adding 6-second pauses at all emotion changes—emphasis points often occur at emotional changes in the speech, so these are the candidate locations for musical underlays.

We then modify the matching table M to align music change points with the speech emphasis points we added in the preprocessing step. To do this, our algorithm first finds the set of change point beats in the music using a novelty-based change point detection algorithm [40, 77]. We examine the music segment on either side of each change point to identify change points that correspond with transitions in emotion labels. Our transition cost table gives a large preference to playing music change points in the output score when emotion changes in music match emotion changes at speech emphasis points. If b_i is a change point whose emotion change matches the emotion change at speech beat k, we increase $M_{j,k}$ by constant $\kappa_{changepoint}$ for all $j \neq i$. This effectively penalizes all other music beats when a change point music beat matches the speech emotion change.

Another strategy expert producers use to emphasize points in the speech is to start or stop the music at those points. To apply this strategy, we penalize music stopping and starting at points other than emotion change boundaries. If speech beat k is not an emotion change boundary, we add cost κ_{exit} to $M_{p_1,k}$ and κ_{enter} to $M_{p_i,k}$ for $i \geq \pi_{min}$. This approach lowers the relative cost for starting and stopping music at emotion changes in the story.

After the algorithm has generated the optimal score, some of the 6 second speech pauses we added in preprocessing may align with pauses in the musical score. Rather than playing six seconds

of silence, our algorithm contracts the final audio, deleting regions where a pause in speech aligns with a pause in the music.

Multiple music tracks. Musical scores often contain segments from multiple music tracks, especially for longer stories. Moreover, most music tracks do not contain every emotion, so using multiple tracks increases the likelihood of matching the output score to the speech emotions.

Figure 5.12 shows how our algorithm combines transition and matching cost tables from two songs. We insert the music beat transition costs from each music track's T along the diagonal (blue squares in Figure 5.12. Music beats can only transition to other beats within the same music track or to pause beats.

After a pause ends, either music track can play (red rectangle, Figure 5.12). We stack the two matching cost tables to create a new matching cost table that covers all possible beats in the musical score (orange rectangles, Figure 5.12). The running time of our algorithm increases linearly in the duration of music added, instead of quadratically, because it does not need to consider inter-music-track transitions.

Setting parameters

We set the main structural parameters based on listening to audio stories: π_{min} of 20 seconds, π_{max} of 35 seconds, δ_{min} of 20 seconds, and δ_{max} of 90 seconds.

Next, we set the distance weights in our optimization empirically, based on the importance of the constraints. We set the acoustic distance parameters $w_t = 1.5$, $w_p = 1.5$, and $w_v = 5$—volume differences are small in magnitude relative to timbre and chroma cosine distance. We set the emotion matching distance parameter $w_{va} = 1$.

Finally, we set the cost parameters: the cost for entering a pause $\kappa_{pause} = 1.4$, empirically less than an audibly "bad" music transition; the intrapause penalty $\kappa_{intra} = .05$, cheaper than all but perfect music transitions; the music start and stop costs—which penalize otherwise free transitions—to small $\kappa_{enter} = \kappa_{exit} = .125$. Lastly, $\kappa_{changepoint} = 1$, which strongly favors inserting musical underlays whenever possible.

Score synthesis

The dynamic programming returns a sequence $\boldsymbol{s}$ of beat and pause indices. We then synthesize the final output score from $\boldsymbol{s}$ by concatenating the audio data for each beat and pause. We ensure smooth-sounding music transitions by inserting crossfades between beats s_i and s_{i+1} if they are not consecutive beats in the original music. In some cases, the musical score becomes out of alignment with the speech because music beats have variable length and we assume a fixed beat length on the speech. Our system corrects this problem every time the music pauses. If a pause segment ends at speech beat k, we start the music again exactly at speech beat k times the average beat duration regardless of the current time in the output. This re-aligns the score with the speech after every pause.

Our system adjusts the volume of the musical score so it is always audible but never overpowers the speech. We control the volume of the output score in two steps. First, we add volume curves

for the score to follow: music fades in for three seconds after music pauses to a low level, increases exponentially to a high-level at speech emphasis points, decreases rapidly to a low level after music solos, and fades out for three seconds as music pauses begin. The second step adjusts these volumes relative to the speech volume. For every segment of the final output where both speech and music are playing, we adjust the decibel level of the music to be 12 dB below the decibel level of the speech. Finally, we notch the 2.76 kHz and 5.63 kHz music frequencies by 6 dB because those frequencies contain important acoustic information for decoding consonants and vowels in speech [65].

Efficient optimization

Because our m and n are small (roughly $n = 500$ beats in a song, and roughly $m = 1000$ beats in the desired output), the entire table can fit in memory and we can use a dynamic programming algorithm to efficiently find this sequence of beats. However, music segment length constraints increase the size of this search space.

If we apply music segment length constraints, our goal is to find the lowest cost sequence of (beat, segment-length) pairs $\boldsymbol{s}'$, using T' and M' in place of T and M in Equation 5.1. The search space at each step has grown from size $(n + \pi_{max})^m$ to size $(n\delta_{max} + \pi_{max})^m$ because we need to consider the state space of all (beat, segment-length) pairs. The running time of a normal dynamic programming algorithm increases quadratically with δ_{max}. However, most of the entries in T' are infinity and are never in an optimal sequence of beats.

We take advantage of these hard constraints to implement an efficient version of this dynamic programming optimization whose runtime increases linearly with δ_{max}. We never explicitly compute T', which can grow prohibitively large. We define our new table T' in terms of blocks from T (see Figure 5.11), so we only store T. We then take advantage of the block structure of T' to limit our search space. For example, if the previous (beat, segment-length) pair in a path is (b_i, d_j), and segment length d_j is less than δ_{min}, our algorithm only needs to minimize over (beat, segment-length) pairs at length d_{j+1} instead of all possible (beat, segment-length) pairs.

Likewise, we do not construct the full table M'. Instead, we have a condition in our algorithm to check if speech bear $t = 1$, which forces the output beat/length pair to have length 1. In other cases, we tile δ_{max} copies of a column in M when our algorithm requires a column of M' during the optimization.

5.6 Interfaces for music editing

We integrated music editing tools in our speech editing interface [77]. The interface contains a *music library* where the producer can import music tracks that she wants to use in the musical score. 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 interfaces for the simple and constrained music retargeting methods.

To lengthen a short music segment, the producer clicks the '↻' button that appears on the music waveform. Our system automatically extends the music by adding a seamless loop to the selected subsegment. The producer can click this button repeatedly as necessary to bring the music to the

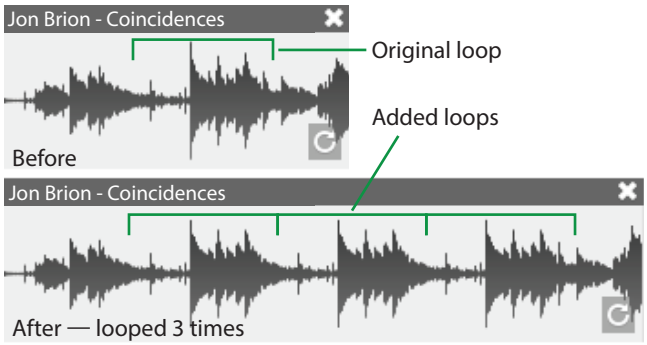

Figure 5.13: A musical segment before and after simple retargeting. Our tool finds a loop in the original segment (before) and in this case the producer repeats the loop three times in the final segment (after).

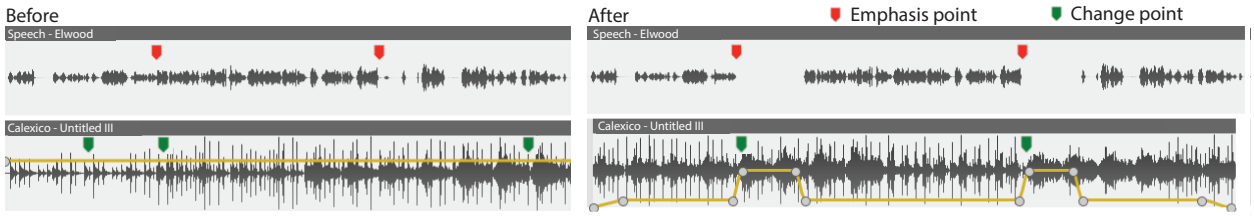

Figure 5.14: In constrained music retargeting, the producer selects multiple speech emphasis points (red markers) and a piece of music (before). Our system automatically identifies music change points (green markers) and aligns them with the emphasis points, by extending or shrinking the music as necessary, while preserving the local beat-level structure of the music (after).

desired length (Figure 5.13). Likewise, if the music is too long, the producer can click the '↺' button to remove audio from the selected subsegment.

Our system also includes functionality to add musical underlays. The producer marks an emphasis point in the speech transcript, selects a song, and then our system automatically produces an underlay. Our system can also produce underlays with multiple change points to emphasize multiple points in speech. The producer initiates this feature by selecting all points in speech that she wants to emphasize (Figure 5.14).

We have yet to integrate emotionally relevant score generation into our speech editing interface. An interface for this integration could be simple, because the algorithm does not require much user input. We can enable the producer to annotate the speech and music emotions (Figure 5.9) within the editing interface, and we can add a new retargeting dialog for full score generation.

5.7 Informal evaluation of the music editor

In our informal user evaluation of our speech editing tools, we also evaluated the usefulness of our music editing tools. These tools impressed the participants, who felt that our simple and constrained retargeting features allow them to rapidly experiment with musical scoring ideas that are otherwise

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."*

5.8 Results

We used our algorithms and editing tools to create edited and scored audio stories. These results include automatically generated musical underlays, stories with edited speech and locally retargeted music, and full-length emotionally relevant musical scores.

5.8.1 Automatic musical underlay evaluation

We created a number of underlays for 3 different audiobook speech clips ("Alice in Wonderland" [20] by Lewis Carroll, "Great Expectations" [29] by Charles Dickens and "Me Talk Pretty One Day" [87] by David Sedaris) representing both male and female voices and a range of emotional tones. We selected the music from tracks commonly heard on the radio documentary program "This American Life" [43]. We have posted these resulting underlays on the website for this research[1].

To evaluate our system's automatically generated underlays, we used a speech clip from David Sedaris's "Me Talk Pretty One Day" audiobook [87] and selected 26 songs that are commonly used on the radio documentary program "This American Life." [43]. For each combination, we found the 3 strongest music change points in each music track using RMS energy features, and automatically generated underlays corresponding to those change points. We asked 3 independent experts who are familiar with "This American Life" to rate the timing, dynamics, and overall quality of each underlay on a 5-point Likert scale (Figure 5.15).

The ratings for the underlays created with the strongest change point correspond to the blue bars in Figure 5.15. With respect to the timing of the change point, the experts gave a rating of 3 (appropriate timing) in 60% of the responses, they rated 89% between 2 (change point arrived slightly too early) and 4 (slightly too late), and 11% as either 1 (far too early) or 5 (far too late). For the dynamics of the music, they rated 59% as 3 (appropriate volume), 94% between 2 (slightly too soft) and 4 (slightly too loud), and 6% as either 1 (far too soft) or 5 (far too loud). To measure overall quality, we asked the experts to evaluate the statement "I am satisfied with the overall quality of the underlay" on a scale from 1 (strongly disagree) to 5 (strongly agree); 50% of the responses were 4 (agree) or higher, 86% were 3 (neutral) or higher, and 14% were 2 (disagree) or lower. On the whole, these ratings suggest that our tool produces high quality underlays with good timing and dynamics.

The ratings for underlays using the second and third strongest musical change points for each song (red and green bars in Figure 5.15, respectively) are similar to the ratings for underlays using the strongest change point. However, the scores for timing and overall quality do decline somewhat as we move from the first to the third strongest change points. This trend suggests that our algorithm for finding a range of change points produces a useful ordering. For dynamics, there is no clear decline in ratings across the three change points. Our tools produce underlays with the appropriate relative dynamics between music and speech regardless of the strength of the change point.

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

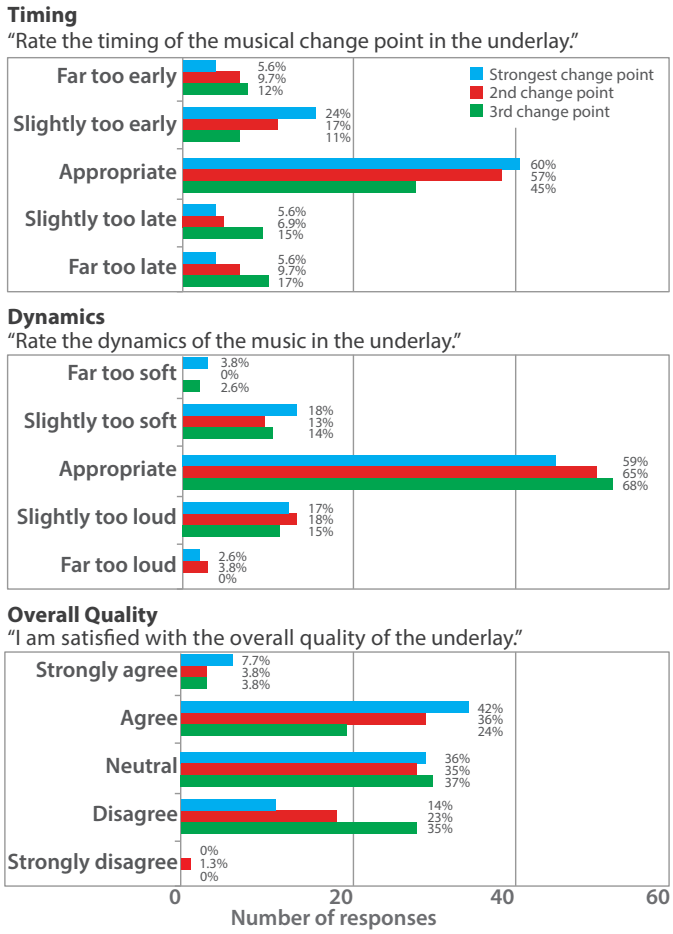

Figure 5.15: Three experts judged the quality of underlays automatically generated with the 3 strongest change points for each of 26 songs. The percentages are the ratios of given ratings to all ratings for each change point strength. While the strongest change point results in the best ratings, the second and third change points also perform well.

5.8.2 Edited stories

We have used our speech and music editing interface to compose seven audio stories from a variety of raw speech sources including scripted narratives, interviews and political speeches. Table 5.1 describes the content of each story. All seven final results as well as the original raw recordings are on our supplemental website[2] for this research.

We 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

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

					Transcript editing usage				Music retargeting usage	
Name	Description	Raw length	Edited length	Total cuts	Text delete	Text copy/paste	View re-takes	Breath insertion	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 5.1: 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 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 the number of 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.

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.

5.8.3 Generated emotionally relevant scores

We have generated 20 musical scores spanning seven music tracks and five speech tracks. We used all three of our emotion labeling techniques: hand-labeling, crowd-labeling, and automatic labeling. Table 5.2 shows our input speech tracks and summarizes the results we have generated, and Table 5.3 details our music tracks. Our supplemental material[3] includes all of these audio results. We labeled the emotions of each story by hand and we had an average of 17.5 crowd workers label each story at an average cost of $4.38 per story. We hand-labeled the music segments for all tracks, and we had an average of 18.5 crowd workers label the segments ($4.63 per track). Our algorithm computes the music track distance tables as a preprocess. On a 2.7 GHz Intel Core i7 processor with 8 GB of RAM, our system takes less than 1 second to find and generate the optimal musical score for a three minute

[3] http://vis.berkeley.edu/papers/emotionscores

Speech tracks					Generated musical scores		
Speech name	Author	Narrator	Dur.	Source	Hand labeled	Crowd Labeled	Auto Labeled
Damon and Pythias	Ella Lyman Cabot	Ginger Cuculo	1:38	LibriVox	**alice**	**alice**, 2815ad, highschool, tub, learn	**alice**
Not That I Care	Molly Reid	James Wood	2:38	NPR	**2815ad**	**2815ad**, alice+timemachine	**2815ad**
Roosts	Zach Brockhouse	Michael Cunningham	3:12	NPR	**tub+highschool**	**tub+highschool**, intriguing+alice	**tub+highschool**
Goldilocks		LearnEnglish Kids	2:10	YouTube	**learn**	**learn**, timemachine	**learn**
The Story of an Hour	Kate Chopin	Matt Bohnhoff	7:20	LibriVox		intriguing+learn	

Table 5.2: We generated 20 scores spanning five different audio stories. The short names on the right correspond to music in Table 5.3. Crowd listeners evaluated the musical scores in bold. A '+' represents a musical score generated with two tracks using our multiple tracks constraint.

Short name	Music track	Artist
alice	Alice Returns	Danny Elfman
2815ad	2815 A.D.	Thomas Newman
highschool	Highschool Lover	Air
learn	You Learn	Jon Brion
tub	The Bathtub	Dan Romer
timemachine	Time Machine	Ryan Miller
intriguing	Intriguing Possibilities	Trent Reznor & Atticus Ross

Table 5.3: We used seven different instrumental music tracks in the musical scores we generated.

story and two three minute music tracks. If we add music segment length constraints with d_{min} of 20 seconds and d_{max} of 90 seconds, the optimization takes 3 minutes. We generated all of our results with d_{min} of 20 seconds and d_{max} of 90 seconds except for "The Story of an Hour" musical score, which we use to demonstrate a longer story with a higher d_{min} (see supplemental material). We also include a basic example of a musical score that we generated for a children's book video.

Figure 5.16 shows a visualization of the twelve of our results—four speech/music combinations with each of the three labeling methods—that correspond to the bold-faced results in Table 5.2. The emotions in the speech and music consistently match in the hand- and crowd-labeled results. In two instances ("Roosts"/crowd and "Not That I Care"/crowd), an incorrect emotion plays in the music for a small number of beats, but neither is audibly noticeable. The automatic results do not match emotions as strongly. In "Not That I Care"/auto and "Goldilocks"/auto, the automatic approach matches large sections of "happy" speech with other music emotions. This happened because the automatic music labeling did not predict any "happy" segments of music, while the automatic speech labeling predicted spans of "happy" that are longer than the maximum length pause π_{max}.

Listening evaluation

To evaluate our generated emotionally relevant scores for stories, we asked crowd workers to listen to and rank four versions of a story—*hand-labeled*, *crowd-labeled*, *automatically labeled*, and *no music*—in terms of overall quality. For each story, twelve US workers, each with over 95% approval rates on Mechanical Turk, evaluated the results. We rejected evaluation tasks if a worker did not

spend enough time on the task to listen to all of the audio stories; we removed 24.6% of the ratings based on this test. Figure 5.16 shows the average ranking for each of the four versions for each speech track. Listeners ranked the results generated using crowd labels (average rank over the four stories of 2.21) as high as hand-labels (average of 2.3). Both methods generated higher quality results than automatic labels (average of 2.52) and the stories with no music (average of 2.95). We find the differences in rankings to be significant ($\chi^2(3) = 9.02$, $p < .05$) using Friedman's nonparameteric test for differences in ranks. Subsequent pairwise comparisons using the Wilcoxon signed rank test find a significant preference for crowd results over no music ($p < .005$) and hand-labeled results over no music ($p < .05$). This result suggests that the "wisdom of crowds" might apply to affective tasks. Our results also show that for all of the speech tracks, our system generates musical scores that improve in overall quality versus the speech without music.

To evaluate the consistency of the emotion labels, we compute Fleiss' and Cohen's kappa values. The average Fleiss' kappa is 0.271 between workers for the speech labels and 0.2 for the music labels. The average Cohen's kappa between our hand-labeled emotions and the most probable crowd labeling is 0.189 for speech and 0.437 for music, which suggests a large difference between the typical crowd-labeling and our expert hand-labeling. Finally, the reliability between our hand-labels and the automatic labels is nearly zero, at kappa of 0.0484 for speech and 0.136 for music. This discrepancy between human and automatic labels is a likely cause of the automatic method's low ranking in our listening evaluation.

5.9 Conclusion

Expertly crafted musical scores can improve the quality and engagement of an audio story, but they are challenging and time-consuming to produce. We have presented algorithms for retargeting music to fit constraints like matching exciting moments in music to emphasis points in speech, and matching emotions in speech to emotions in music. Our music editing interfaces allow producers to specify these high-level goals without manipulating music waveforms and without needing audio editing or music composition expertise.

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.

While we successfully generated emotionally relevant musical scores, labeling emotions is a difficult and ill-defined task. We focused on four emotions that may not be universally understandable or applicable. Our results may more closely map to how listeners experience emotions if instead of focusing on labeling individual emotions in isolation, we focus on understanding emotional transitions. For instance, how does a listener's emotional understanding of a piece of music change throughout the song, and can we match that to a story arc? This approach is reminiscent of Luke Howard's idea to classify the transitional forms of clouds [49].

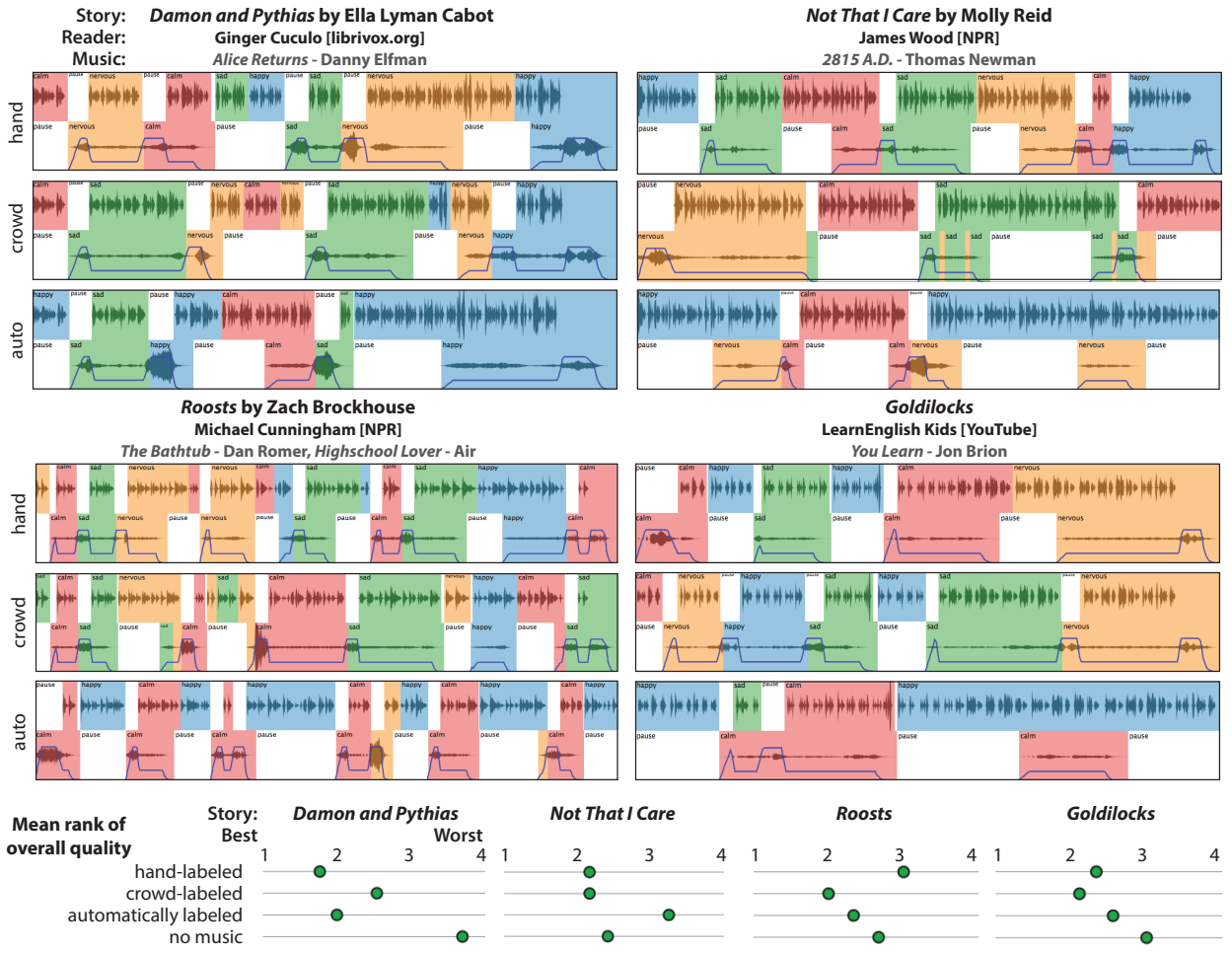

Figure 5.16: We visualize twelve of our generated results above. We generated each speech/music pair using all three of our labeling methods. Evaluators on Mechanical Turk listened to three different scores for a speech track and the original speech without music. These charts show the average ranking in overall quality of the four clips among the evaluators.

Chapter 6

Integrity of manipulated audio

The goal throughout our research is to improve the process of creating audio stories by allowing producers to work with audio at a higher semantic level. Our interfaces and algorithms, like most media-editing tools, enable producers to make edits that fail to preserve the original meaning or intent of the media. While these manipulations are an integral part of our tools, we acknowledge that they allow producers to lie and misrepresent the original sources.

6.1 Integrity in audio story production

In non-fiction audio production, like any reporting, there are competing goals that the producer tries to meet.

6.1.1 Content stakeholders

Each of the individual pieces of media used in an audio story has stakeholders: people invested in or concerned with the original intent of the content. Using media editing tools, the producer has the power to allow or deny stakeholders their desired outcomes. The stakeholders include:

- The people involved in a story—whether interviewed directly or referenced by other people or narrators in a story—have a stake in their lives, thoughts, and feelings being faithfully represented. The producer could censor the interviewees or take their quotes out of context, thus stripping them of their rights to free speech and fair representation.
- The storytellers, narrators, and producers of a story have stakes in how the listeners perceive the story, wanting to convey a certain set of ideas, facts, and opinions. The producers may have artistic visions for the story that may be in conflict with the other stakeholders in the story. In the case that these ideas and opinions are not supported by the collected facts and interviews, the producers could misrepresent the truth.

- The artists of music used in a story have a stake in how listeners perceive their music, both structurally (e.g., their songs should play in their entirety) and emotionally (e.g., the songs should evoke a certain emotion).
- Finally, the media rightsholders want the producer to use their media in accordance with applicable licenses and copyrights. The producer could use and remix/retarget sampled audio footage without obtaining permission from its authors.

While non-fiction audio stories involve all of these stakeholders, fictional audio stories (e.g., radio dramas or fictional audio books) must still consider the desires of the musicians and rightsholders. In the case of an audio book, the narrator is often not the book's author. The author has a stake in how the narrator portrays the book.

6.1.2 Goals for audio integrity

Regardless of the editing tools used, the producer should strive to maintain the desires of the stakeholders. At a basic level, the producer should preserve digital rights management and copyright of any media used, and respect fair use where applicable. As a person creating a widely disseminated record of a person, place, or event, the producer should also strive for truth in reporting.

In trying to maintain the desires of stakeholders and the ideals of truth in reporting and proper rights management, the producer may find that, for example, the stakeholders have unaligned interests, or that their interests are in flux, changing over time. The producer should, as much as possible, disclose these conflicts in the finished production. All stakeholders have the right to question the integrity of the final auditory experience.

6.2 Integrity in our tools

The above considerations apply to non-fiction audio story production in general. In this section, we investigate how our tools do and do not support these integrity guidelines.

6.2.1 Audio manipulation

Our recording and editing tools enable different kinds of manipulation. Each of these manipulations has the potential to alter the original meaning of the content.

- *Narration Coach* allows the user to resynthesize speech, changing its speed and its emphasized words. This can change the meaning of the speech (e.g., changing the emphasized word in the sentence, "I never said she stole my money" results in a different meaning).
- Our speech editor allows the producer to copy, paste, and delete speech at the word level. The producer can create new sentences and change the meaning or context of existing sentences.

- More subtly, our speech editor allows the producer to change the cadence of speech by adding and removing breaths and pauses. A long pause can, for example, suggest tension or uncertainty that was not present in the original speech recording.
- Our music editing tools can retarget songs, changing their structure. The resulting "remixed" music may not resemble the original track.
- Using music to score speech can change the perceived meaning of the music and the speech, even if the music and speech themselves are not edited.

These manipulations are similar to the kinds of edits that designers, producers, and DJs currently perform using software like Adobe Photoshop, Avid Pro Tools, and Ableton Live. Existing work on "remix culture" explores the legal and ethical ramifications of the products of these editing tools [38, 59], and helps creators decide whether their creations qualify as fair use or require licensing [50].

6.2.2 Automation and delegation

Most of the tools in our research aim to speed up production by automating parts of the recording and editing pipeline. *Narration Coach* helps producers organize recordings while automatically analyzing recorded takes to help create better recordings. The speech editor replaces the traditional low-level waveform edits in DAWs with quicker, word- or sentence-level edits. Our music editing algorithms automatically analyze and re-sequence music to meet the producer's constraints.

After using these new automated tools, the producer may become reluctant to perform these tasks using older, time-consuming methods. Instead, the producer may delegate to the automation, reverting to the manual methods only when the automation fails. This delegation can be advantageous, allowing a better use of human effort: the producer can deal with media faster at a more semantically meaningful level (e.g., words, rather than waveforms). However, excessive automation can lead to a loss of agency and control in the recording and editing process.

Parasuraman et al. [69] propose a model for designers to determine the appropriate level of automation in a system. This model describes four stages that a system can automate: information acquisition, information analysis, decision selection, and action implementation. The primary criteria for selecting the desirable level of automation in these stages is that of human performance consequences. That is, how will the automation impact the user's mental workload, situation awareness, complacency, and skill degradation. Our audio production tools provide automation mainly in the information analysis (e.g., predicting emphasized words; finding music change points; aligning text and speech) and decision selection (e.g., finding an optimal retargeting of music to fit the speech's emotions). In our tools, the users are responsible for information acquisition (e.g., providing the input speech and music footage) and action implementation (e.g., determining whether to use and publish a proposed underlay or musical score). None of our tools provide complete automation where the system executes irreversible actions on the user's behalf. In this sense, our tools give users the opportunity to stay in the loop and maintain agency throughout the audio story production process.

Transparency in automation

Ultimately, the producer makes these decisions about automation and agency; our tools do not force or prohibit certain forms or automation. Our tools can, however, make automation transparent so the producer fully understands the edits and manipulations she is making.

Edit log: Our tools perform nondestructive edits and maintain their current state as internal metadata. This metadata, a kind of EDL (edit decision list), fully describes how to recreate the final audio from the original sources. Because our edits are nondestructive, we also retain provenance of the original sources in their individual metadata (e.g., the date and time of recording).

Visual cues: While the producer can inspect the internal metadata, our tools also provide static visual cues to help her track the changes she makes. Our speech editor displays red bars in the text transcript between non-contiguous audio segments. Our music editing tools likewise show red bars in the music waveforms where the music differs from its original progression. These cues are static, but we could also show a step-by-step, animated history of editing sessions.

Injurious automation

Even with these techniques for adding transparency in automation, the producer may create content that proves harmful to some or all of the stakeholders. In the event of this harm, who is responsible: the producer, or the designers and engineers of the automation tools? Furthermore, suppose that the producer provides constraints to the automation such that the system can create multiple valid outputs, each of which causes harm to a subset of the stakeholders. Which of the injurious output options should the system choose to produce?

These kinds of ethical question are central in the ongoing debate on regulating autonomous vehicles (AVs, i.e., "self-driving cars"). AVs promise to automate transportation, and travelers will delegate to this automation. If an autonomous vehicle hits a pedestrian, who is responsible for the damages: the vehicle's passengers, its owner, its manufacturer, or another party? That responsible party may differ depending on what type of vehicle it is (e.g., a passenger car versus a bus versus a police car). AVs may have no choice but to injure people in certain situations, like in the ethical "Trolley problem" thought experiment [39]. In these cases, how should the vehicles decide whom to injure [17]?

While this example is extreme in that media editing tools will likely not cause physical damage, these tools can, however, cause psychological, social, economic, and legal harm. The ethical questions about responsibility and harm are not yet codified in laws and regulations for autonomous vehicles or for algorithmic automation in general. As a society, we need to find satisfactory answers to these ethical questions as we move further into an automated world.

6.3 Conclusion

Every piece of media used in a composing a non-fiction audio story has rightsholders and stakeholders. A producer can intentionally manipulate media in ways that violate its rights or conflict

with the ideals of its stakeholders. The producer must be cognizant of these rights and stakes as she records and edits her story.

Our speech and music recording and editing tools do not prohibit the producer from violating the rights and desires of the media stakeholders. However, the tools record metadata that enable the producer to share the exact nature of her edits. If a stakeholder takes issue with the integrity of the auditory experience, the producer and the stakeholder can use the metadata to discuss whether the edits are acceptable. Rather than trying to take a definitive stance on what is and what is not allowed in media editing, our tools give stakeholders the resources required to argue their side. However, even with these tools, the producer must understand that our automated systems have the potential to cause harm.

Chapter 7

Future work

In our research, we provide tools for the recording and editing phases of audio story production. We believe there are interesting challenges in helping producers brainstorm and plan stories, creating alternative versions of stories for different audiences, and producing other kinds of multimedia content like videos.

7.1 Audio story ideation

None of our tools address the ideation phase of storytelling, where the producer brainstorms, researches, and writes a story, and when she conducts interviews. Research systems like Motif [55] and mobile applications like Directr [30] help novices construct video stories using storyboards and templates. We could extend our pipeline with tools like these. By analyzing a large collection of audio stories, we could create templates for common, engaging stories that the producer could use while constructing her story. We could also extend *Narration Coach*, our active capture speech recording tool, to provide useful feedback for producers as they record interviews.

For example, a template for introducing a character could provide audio examples from high quality stories, suggest a set of questions for the producer to ask the interviewee in the field, and show the producer different patterns for splicing narration in with interview footage. We could also ensure that the producer includes a good balance of narration and interview footage, rather than leaning too heavily on one or the other. In building these tools, we would need to make sure that they are flexible and customizable to prevent producers from creating stories that are too similar to each other.

7.2 Adaptive content

Podcasts are often long, unedited conversations. At the time of writing, the average length of the top ten podcast episodes on iTunes was over 90 minutes. While these may appeal to people with long chunks of time to fill, some listeners do not have as much listening time and struggle to keep up with a backlog of long episodes. Some listeners may also be more interested in certain segments

of long podcasts and less interested in other parts. We could design tools and algorithms to create shorter versions of the story that maintain the semantic meaning and key moments of the story and interviews. We could also create versions that focus on specific topics of interest to the listener while removing information about a less interesting topic.

One way we could create these adaptive stories is by having the user annotate semantic dependencies in the story, from which we can induce a directed acyclic graph on small content segments. We would topologically sort the graph based on the original progression of the story. Then, we could compose a shorter version of the story by selecting a subset of the nodes such that for every node in the subset, the node's ancestors are also in the subset. The user could also provide content tags on the nodes, which we could use to compose stories that covered certain topics. To speed up this composition process, we could apply machine learning and natural language processing techniques to predict the semantic dependency graph and the content tags. While users may prefer these customized audio stories, we may be removing an important social aspect of the stories, i.e., listeners who have consumed the same content can discuss it.

7.3 Video production

Our work focuses on creating audio stories. However, more people share video than audio. We could extend the techniques that we developed in our research to video capture and editing. Some of our techniques have natural extensions to video. For example, we could generate musical scores for video without any changes to our algorithm. However, musical scores in video follow different design guidelines than in audio stories, so we would need to study these differences in order to modify our structural and stylistic constraints.

Other techniques would require less trivial extensions for video. Our narration recording tool would need to incorporate additional timing constraints in order to be useful for video narrations (e.g., "talking head" video footage). If the producer wanted to use the tool to guide video capture instead of just narration, we would need to incorporate computer vision techniques to detect visual issues like bad lighting and poor composition. The resulting application might behave like a combination of *Narration Coach* and Carter et al.'s [21] NudgeCam. First, an expert producer trains the system how to capture a certain type of shot. The system would analyze the training data to determine the characteristic audio/visual properties of the shot. Finally, while the user records, the system analyzes the footage and suggest ways to fix problems.

Chapter 8

Conclusion

In this work, we have presented algorithms and interfaces that address three phases of the audio story production pipeline.

In Chapter 3 (Recording speech) we developed *Narration Coach*, a system for helping novice voiceover actors record and improve narrations. To create *Narration Coach*, we studied voiceover performance guidelines. *Narration Coach* organizes a recording session by aligning the user's recordings to a script. Our tool then analyzes the user's speech to provide text-based feedback on common narration problems, and also provides resyntheses for problems in emphasis and flow.

In a pilot study, we found that novice voiceover actors created better narrations when using *Narration Coach* than when using Adobe Audition, a traditional DAW. One limitation of our approach is its reliance on automatic speech recognition, which can be inaccurate with some speakers. Another limitation of *Narration Coach* is that by pushing all speakers to conform to certain guidelines, the resulting audio may lose the individualism of unaided narrations.

Next, in Chapter 4 (Editing speech) we designed and built an interface for editing speech at a high semantic level, using words rather than waveforms. Our speech editor allows the producer to edit speech using word-level cut, copy, paste, and delete operations, as in a word processor. The editor also provides tools to add, remove, and adjust breaths and pauses, which help maintain a natural cadence in the edited speech. The producer can import multi-tracked interview footage, and our tool automatically keeps the tracks in sync as the producer makes edits. We found promising results from an informal evaluation of our speech editing system. Producers preferred this text-based method of editing to the traditional waveform-based method. The main limitation of the speech editor is that it requires verbatim transcripts of the speech tracks. Inaccurate transcripts lead to alignment errors, reducing the effectiveness of the word-level editing tools.

In Chapter 5 (Editing music) we developed an efficient framework for retargeting music for audio stories. With this framework, we could create looped musical segments, add musical underlays to emphasize key moments in speech, and generate full-length, emotionally relevant scores that satisfy stylistic and structural constraints. We integrated the looping and musical underlay tools into our speech editing interface. In an informal evaluation, we found that producers appreciated the music editing tools in the speech editor, as they allowed the producers to quickly iterate on scoring ideas. Impartial listeners found our emotionally relevant scores to be of high quality. One limitation

of our retargeting framework is that it requires that the music have detectable beats. We designed our framework for instrumental music; it is agnostic to lyrics. This can result in confusing or meaningless lyrics in retargeted music.

Finally, in Chapter 6 (Integrity of manipulated audio) we outlined the stakeholders involved in the production and distribution of audio stories and discussed how our recording and editing tools can manipulate original sources. We described how our tools record edits as metadata, allowing the producer to understand and argue for how she is modifying the original audio.

All of these algorithms and interfaces aim to improve audio story production. The central approach in our research is to first understand how experts perform production tasks, then to use that information to inform algorithms and interfaces that automate those procedures. Our tools provide the producer a means of recording and editing audio stories at a higher semantic level than in DAWs like Audition or Pro Tools. While our tools automate low-level procedures, they all require human input. For example, the producer selects the speech emphasis point and music track when adding musical underlays, and she can select the emotion labels and music tracks for generating emotionally relevant scores. By adding the user-in-the-loop, we believe that our tools offer the producer stylistic control while speeding up tedious and difficult operations in audio story production, ultimately leading to a better audio story production experience.

Bibliography

[1] Hindenburg Journalist Pro. http://hindenburgsystems.com. Accessed: 2015-12-13.

[2] Stupid fade tricks. http://transom.org/?p=7543, September 2003. Accessed: 2015-12-13.

[3] Third coast audio library. http://www.thirdcoastfestival.org/library, June 2015. Accessed: 2015-12-13.

[4] Abel, J., and Glass, I. *Radio: An Illustrated Guide*. WBEZ Alliance Inc., 1999.

[5] 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.

[6] Abumrad, J., and Krulwich, R. Radiolab. http://www.radiolab.org/, June 2015. Accessed: 2015-12-13.

[7] Agrawala, M., Li, W., and Berthouzoz, F. Design principles for visual communication. *Communications of the ACM (CACM) 54*, 4 (Apr. 2011), 60–69.

[8] Arikan, O., Forsyth, D. A., and O'Brien, J. F. Motion synthesis from annotations. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 402–408.

[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] Berry, R., Makino, M., Hikawa, N., and Suzuki, M. The augmented composer project: The music table. In *Proc. of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, IEEE Computer Society (2003), 338.

[11] Berthouzoz, F., Li, W., and Agrawala, M. Tools for placing cuts and transitions in interview video. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 67.

[12] Biewen, J. The transom review: John Biewen. http://transom.org/?p=19659, September 2011. Accessed: 2015-12-13.

[13] Bischof, M., Conradi, B., Lachenmaier, P., Linde, K., Meier, M., Pötzl, P., and André, E. Xenakis: combining tangible interaction with probability-based musical composition. In *Proc. of the 2nd international conference on Tangible and embedded interaction*, ACM (2008), 121–124.

[14] Black, A. W., and Taylor, P. A. The Festival Speech Synthesis System: System documentation. Tech. Rep. HCRC/TR-83, Human Communciation Research Centre, University of Edinburgh, Scotland, UK, 1997. Avaliable at http://www.cstr.ed.ac.uk/projects/festival/.

[15] Blu, S., Mullin, M. A., and Songé, C. *Word of Mouth: A Guide to Commercial and Animation Voice-over Excellence*. Silman-James Press, 2006.

[16] Boersma, P., and Weenink, D. Praat, a system for doing phonetics by computer. Available at http://www.praat.org/.

[17] Bonnefon, J., Shariff, A., and Rahwan, I. Autonomous vehicles need experimental ethics: Are we ready for utilitarian cars? *CoRR abs/1510.03346* (2015). Available at http://arxiv.org/abs/1510.03346.

[18] Calvo, R. A., and D'Mello, S. Affect detection: An interdisciplinary review of models, methods, and their applications. *Affective Computing, IEEE Transactions on 1*, 1 (2010), 18–37.

[19] Carrascal, J., and Jordà, S. Multitouch interface for audio mixing. In *Proceedings of New Interfaces for Musical Expression (NIME)* (2011).

[20] Carroll, L. Alice's Adventures in Wonderland, read by Kristen McQuillin. https://librivox.org/alices-adventures-in-wonderland-by-lewis-carroll/, Jan. 2006. Accessed: 2015-12-13.

[21] Carter, S., Adcock, J., Doherty, J., and Branham, S. Nudgecam: Toward targeted, higher quality media capture. In *Proceedings of the International Conference on Multimedia (MM)*, ACM (New York, NY, USA, 2010), 615–618.

[22] 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.

[23] CastingWords. https://castingwords.com, Dec. 2015. Accessed: 2015-12-13.

[24] Chang, A. R., and Davis, M. Designing systems that direct human action. In *Extended Abstracts on Human Factors in Computing Systems (CHI)*, ACM (2005), 1260–1263.

[25] Dannenberg, R. B. An intelligent multi-track audio editor. In *Proceedings of International Computer Music Conference (ICMC)*, vol. 2 (2007), 89–94.

[26] Davis, M. Active capture: integrating human-computer interaction and computer vision/audition to automate media capture. In *International Conference on Multimedia and Expo (ICME)*, vol. 2, IEEE (2003), II–185–II–188.

[27] Davis, M. Editing out video editing. *IEEE MultiMedia 10*, 2 (2003), 54–64.

[28] Design, S. S. Voice lesson 2: Marking a script. https://www.youtube.com/watch?v=LwS7WD7WQ3Y. Accessed: 2015-03-31.

[29] Dickens, C. Great Expectations, read by Mark F. Smith. http://librivox.org/great-expectations-by-charles-dickens/, Dec. 2008. Accessed: 2015-12-13.

[30] Directr. https://itunes.apple.com/us/app/directr-simple-powerful-video/id526717506, Feb. 2015. Accessed: 2015-12-13.

[31] Dolson, M. The phase vocoder: A tutorial. *Computer Music Journal 10*, 4 (1986), 14–27.

[32] Drew, P. Take your copy to the woodshed. http://www.peterdrewvo.com/html/analyze_the_copy_first.html, 2005. Accessed: 2015-03-31.

[33] Dusterhoff, K., and Black, A. W. Generating Fo contours for speech synthesis using the tilt intonation theory. In *Intonation: Theory, Models and Applications* (1997).

[34] El Ayadi, M., Kamel, M. S., and Karray, F. Survey on speech emotion recognition: Features, classification schemes, and databases. *Pattern Recognition 44*, 3 (2011), 572–587.

[35] Ellis, D. P., and Poliner, G. E. Identifying cover songs with chroma features and dynamic programming beat tracking. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 4, IEEE (2007), IV–1429–IV–1432.

[36] Farbood, M., Pasztor, E., and Jennings, K. Hyperscore: a graphical sketchpad for novice composers. *IEEE Computer Graphics and Applications 24*, 1 (2004), 50–54.

[37] Fazekas, G., and Sandler, M. Intelligent editing of studio recordings with the help of automatic music structure extraction. *Audio Engineering Society Convention* (2007).

[38] Ferguson, K. Everything is a remix. http://everythingisaremix.info/watch-the-series/. Accessed: 2015-08-11.

[39] Foot, P. The problem of abortion and the doctrine of the double effect. *Oxford Review 5* (1967).

[40] Foote, J. Automatic audio segmentation using a measure of audio novelty. In *International Conference on Multimedia and Expo (ICME)*, vol. 1, IEEE (2000), 452–455.

[41] Foote, J., Cooper, M., and Girgensohn, A. Creating music videos using automatic media analysis. In *Proceedings of the tenth ACM international conference on Multimedia (MM)*, ACM (2002), 553–560.

[42] 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.

[43] Glass, I. This American Life. http://www.thisamericanlife.org/. Accessed: 2015-12-13.

[44] Glass, I. The transom review: Ira Glass. http://transom.org/?p=6978, June 2004. Accessed: 2015-12-13.

[45] Goldberg, D. *The Voice Over Technique Guidebook with Industry Overview*. Edge Studio, 2010.

[46] Grossman, T., Fitzmaurice, G., and Attar, R. A survey of software learnability: metrics, methodologies and guidelines. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, ACM (2009), 649–658.

[47] Heer, J., Good, N. S., Ramirez, A., Davis, M., and Mankoff, J. Presiding over accidents: system direction of human action. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, ACM (2004), 463–470.

[48] Hindenburg ABC. http://hindenburg.com/products/hindenburg-abc/. Accessed: 2015-12-13.

[49] Howard, L. *Essay on the Modifications of Clouds*. 1865.

[50] Jackson, B. A handy guide to remixing without getting sued. http://www.buzzfeed.com/benjaminj4/infographic-how-to-remix-without-getting-sued, Oct. 2012. Accessed: 2015-08-11.

[51] Jehan, T. *Creating music by listening*. PhD thesis, Massachusetts Institute of Technology, 2005.

[52] Jehan, T., and Sundram, J. The EchoNest remix earworm. https://github.com/echonest/remix/tree/master/examples/earworm, Apr. 2013. Accessed: 2013-04-02.

[53] Jiang, D.-N., Lu, L., Zhang, H.-J., Tao, J.-H., and Cai, L.-H. Music type classification by spectral contrast feature. In *International Conference on Multimedia and Expo (ICME)*, vol. 1, IEEE (2002), 113–116.

[54] Kawahara, H., Morise, M., Takahashi, T., Nisimura, R., Irino, T., and Banno, H. Tandem-straight: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, fo, and aperiodicity estimation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE (2008), 3933–3936.

[55] Kim, J., Dontcheva, M., Li, W., Bernstein, M. S., Steinsapir, D., and Research, A. Motif: Supporting novice creativity through expert patterns. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, ACM (2015), 1211–1220.

[56] Kim, Y. E., Schmidt, E. M., Migneco, R., Morton, B. G., Richardson, P., Scott, J., Speck, J. A., and Turnbull, D. Music emotion recognition: A state of the art review. In *Proceedings of International Society for Music Information Retrieval Conference (ISMIR)* (2010), 255–266.

[57] Kurihara, K., Goto, M., Ogata, J., Matsusaka, Y., and Igarashi, T. Presentation sensei: A presentation training system using speech and image processing. In *Proceedings of the 9th International Conference on Multimodal Interfaces (ICMI)*, ACM (New York, NY, USA, 2007), 358–365.

[58] Lasecki, W. S., Murray, K. I., White, S., Miller, R. C., and Bigham, J. P. Real-time crowd control of existing interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST)*, ACM (2011), 23–32.

[59] Lessig, L. *Remix: Making art and commerce thrive in the hybrid economy*. Penguin Press HC, The, 2008.

[60] Li, F. C., Gupta, A., Sanocki, E., He, L.-w., and Rui, Y. Browsing digital video. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, vol. 1, ACM (2000), 169–176.

[61] Logan, B. Mel frequency cepstral coefficients for music modeling. In *Proceedings of International Society for Music Information Retrieval Conference (ISMIR)* (2000).

[62] 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.

[63] Mauch, M., and Dixon, S. pyin: A fundamental frequency estimator using probabilistic threshold distributions. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2014). in press.

[64] McFee, B., and Ellis, D. P. Learning to segment songs with ordinal linear discriminant analysis. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE (2014).

[65] McKee, J. Using music: The kitchen sisters. http://transom.org/2014/using-music-the-kitchen-sisters/, Feb. 2014. Accessed: 2015-12-13.

[66] Monteith, K., Francisco, V., Martinez, T., Gervas, P., and Ventura, D. Automatic generation of emotionally-targeted soundtracks. In *Proceedings of the Second International Conference on Computational Creativity* (2011), 60–62.

[67] Monteith, K., Martinez, T. R., and Ventura, D. Automatic generation of music for inducing emotive response. In *Proceedings of the First International Conference on Computational Creativity* (2010), 140–149.

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

[69] Parasuraman, R., Sheridan, T. B., and Wickens, C. D. A model for types and levels of human interaction with automation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on 30*, 3 (2000), 286–297.

[70] Posner, J., Russell, J. A., and Peterson, B. S. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and psychopathology 17*, 03 (2005), 715–734.

[71] Robertson, H. 10 ways to build your voice-over skills. http://www.videomaker.com/article/15804-10-ways-to-build-your-voice-over-skills, Nov. 2013. Accessed: 2015-03-31.

[72] Rodgers, J. *The Complete Voice & Speech Workout: 75 Exercises for Classroom and Studio Use*. Hal Leonard Corporation, 2002.

[73] Roma, G., and Xambó, A. A tabletop waveform editor for live performance. In *Proceedings of New Interfaces for Musical Expression (NIME)* (2008).

[74] Rosenberg, A. Autobi-a tool for automatic tobi annotation. In *INTERSPEECH* (2010), 146–149.

[75] Rubin, S., and Agrawala, M. Generating emotionally relevant musical scores for audio stories. In *Proceedings of the 27th annual ACM symposium on User interface software and technology (UIST)*, ACM (2014), 439–448.

[76] Rubin, S., Berthouzoz, F., Mysore, G., Li, W., and Agrawala, M. Underscore: musical underlays for audio stories. In *Proceedings of the 25th annual ACM symposium on User interface software and technology (UIST)*, ACM (2012), 359–366.

[77] Rubin, S., Berthouzoz, F., Mysore, G. J., Li, W., and Agrawala, M. Content-based tools for editing audio stories. In *Proceedings of the 26th annual ACM symposium on User interface software and technology (UIST)*, ACM (2013), 113–122.

[78] Rubin, S., Mysore, G. J., Berthouzoz, F., and Agrawala, M. Capture-time feedback for recording scripted narration. In *Proceedings of the 28th annual ACM symposium on User interface software and technology (UIST)*, ACM (2015), 191–199.

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

[80] Russell, J. A. A circumplex model of affect. *Journal of personality and social psychology 39*, 6 (1980), 1161.

[81] Saragih, J. M., Lucey, S., and Cohn, J. F. Deformable model fitting by regularized landmark mean-shift. *International Journal of Computer Vision 91*, 2 (2011), 200–215.

[82] Schmidt, E. M., and Kim, Y. E. Prediction of time-varying musical mood distributions from audio. In *Proceedings of International Society for Music Information Retrieval Conference (ISMIR)* (2010), 465–470.

[83] Schmidt, E. M., and Kim, Y. E. Modeling musical emotion dynamics with conditional random fields. In *Proceedings of International Society for Music Information Retrieval Conference (ISMIR)* (2011), 777–782.

[84] Schödl, A., Szeliski, R., Salesin, D. H., and Essa, I. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co. (2000), 489–498.

[85] Schwarz, D. The caterpillar system for data-driven concatenative sound synthesis. In *Proceedings of Digital Audio Effects (DAFx)* (2003), 135–140.

[86] Schwarz, D., et al. A system for data-driven concatenative sound synthesis. In *Proceedings of Digital Audio Effects (DAFx)* (2000), 97–102.

[87] Sedaris, D. *Me talk pretty one day*. audiobook. Little, Brown and Company, 2000.

[88] Silverman, K. E., Beckman, M. E., Pitrelli, J. F., Ostendorf, M., Wightman, C. W., Price, P., Pierrehumbert, J. B., and Hirschberg, J. Tobi: a standard for labeling english prosody. In *Proceedings of International Conference on Spoken Language Processing (ICSLP)*, vol. 2 (1992), 867–870.

[89] Simon, I., Morris, D., and Basu, S. MySong: automatic accompaniment generation for vocal melodies. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, ACM (2008), 725–734.

[90] Smaragdis, P. User guided audio selection from complex sound mixtures. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology (UIST)*, ACM (2009), 89–92.

[91] Tauscher, J.-P., Wenger, S., and Magnor, M. A. Audio resynthesis on the dancefloor: A music structural approach. In *Proceedings of the Vision, Modeling, and Visualization Workshop (VMV)*, Eurographics Association (2013), 41–48.

[92] Taylor, P. *Text-to-speech synthesis*. Cambridge university press, 2009.

[93] Valbret, H., Moulines, E., and Tubach, J. Voice transformation using psola technique. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1 (1992), 145–148 vol.1.

[94] Warriner, A. B., Kuperman, V., and Brysbaert, M. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods 45*, 4 (2013), 1191–1207.

[95] Wenger, S., and Magnor, M. A genetic algorithm for audio retargeting. In *Proceedings of the 20th ACM international conference on Multimedia (MM)*, ACM (2012), 705–708.

[96] Wenner, S., Bazin, J.-C., Sorkine-Hornung, A., Kim, C., and Gross, M. Scalable music: Automatic music retargeting and synthesis. In *Computer Graphics Forum*, vol. 32, Wiley Online Library (2013), 345–354.

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

[98] Whittaker, S., and Amento, B. Semantic speech editing. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, vol. 24, ACM (2004), 527–534.

[99] Wilcox, J. *Voiceovers: Techniques and Tactics for Success*. Allworth Press, 2007.

[100] Willett, W., Ginosar, S., Steinitz, A., Hartmann, B., and Agrawala, M. Identifying redundancy and exposing provenance in crowdsourced data analysis. *Visualization and Computer Graphics, IEEE Transactions on 19*, 12 (2013), 2198–2206.

[101] Wright, J., Oppenheim, D., Jameson, D., Pazel, D., and Fuhrer, R. Cyberband: A "hands-on" music composition program. In *Proceedings of International Computer Music Conference (ICMC)* (1997), 383–386.

[102] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., Moore, G., Odell, J., Ollason, D., Povey, D., et al. *The HTK book*, vol. 2. Entropic Cambridge Research Laboratory Cambridge, 1997.

[103] 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).

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

[105] Zils, A., and Pachet, F. Musical mosaicing. In *Proceedings of Digital Audio Effects (DAFx)*, vol. 2 (2001).

Appendix A

Coffee

In addition to learning about conducting research, writing papers, and giving presentations, I learned the wonders of coffee. These Americanos, Gibraltars, and pourovers are a welcome departure from the Mountain-Dew-centric caffeination techniques of my younger self.

A.1 Cafe log, May—December 2015

As an homage to coffee and its ever-positive impact on my work, here are the specialty coffee shops at which I worked on my dissertation. Within cities, I roughly sort these from most-visited to least-visited.

Berkeley: Babette, Artis, Allegro Coffee Roasters, and Philz.

Seattle: Milstead & Co., Vif Wine|Coffee, Slate Coffee, Trabant Coffee & Chai, Neptune Coffee, Ballard Coffee Works, Fremont Coffee, Elm Coffee Roasters, Cafe Ladro (Fremont), Cafe Ladro (Downtown), Victrola Coffee and Art, Porchlight Coffee and Records, Espresso Vivace (Alley 24), and Zoka Coffee (University Village).

San Francisco: Workshop Cafe.

The following are the other shops I visited during the time span of writing my dissertation. While I didn't work on my dissertation at these shops, sometimes you have to stop and smell the espresso.

Berkeley: Alchemy Collective Cafe, Highwire (née Local 123, San Pablo), and Equator.

Albany, CA: Highwire (née Local 123, Flowerland).

Emeryville, CA: Scarlet City Espresso Bar.

Oakland, CA: Trouble Coffee Co.

Mill Valley, CA: Equator.

San Francisco: Ritual (Hayes Valley), Jane (Fillmore), Artis (Hayes Valley), Special Xtra, Blue Bottle (Hayes Valley), Wrecking Ball Coffee Roasters, Mazarine Coffee, Red Door Coffee, Stanza Coffee, Coffee Cultures, Four Barrel (Valencia), The Mill, The Richfield, Blue Bottle (Mint Plaza), Blue Bottle (Ferry Building), and Paramo Coffee.

Seattle: Caffe Vita (Airport), Tougo Coffee Co., Tin Umbrella Coffee, and Analog Coffee.

Portland, OR: Good Coffee (SE Division), Good Coffee (SE Salmon), Barista (Nob Hill), Stumptown (SE Division), Courier Coffee, and Heart Coffee (Westside).

Vancouver, WA: Torque Coffee.

Eugene, OR: Tailored Coffee Roasters.

Houston: Blacksmith, The Honeymoon Cafe and Bar, Boomtown Coffee, and Southside Espresso.

Matagalpa, Nicaragua: Selección Nicaragüense.

Jersey City, NJ: Dame Espresso Bar.

A.2 Recommendations

I make no claim to my coffee knowledge, but my hope is that the listing of cafes I've visited provides a context for my recommendations.

For the best overall coffee experience, go to the Ballard location of **Slate Coffee** in Seattle and order the tasting menu. You will drink four or five courses of the best coffee, espresso, and milk you have ever tasted. **Vif Wine|Coffee** in Seattle and **Babette** in Berkeley are the best places to work. They are beautiful cafes with friendly baristas, delicious coffee, and ample seating. For other outstanding cups of coffee and espresso drinks, I recommend **Ritual** in San Francisco, **Milstead & Co.** in Seattle, and the aptly named **Good Coffee** in Portland, Oregon.