TypeTalker: A Speech Synthesis-Based Multi-Modal Commenting System

Ian Arawjo
Cornell University
Ithaca, USA
iaa32@cornell.edu

Dongwook Yoon
Cornell University
Ithaca, USA
dy252@cornell.edu

François Guimbretière
Cornell University
Ithaca, USA
francois@cs.cornell.edu

ABSTRACT
Speech commenting systems have been shown to facilitate asynchronous online communication from educational discussion to writing feedback. However, the production of speech comments introduces several challenges to users, including overcoming self-consciousness and time consuming editing. In this paper, we introduce TypeTalker, a speech commenting interface that presents speech as a synthesized generic voice to reduce speaker self-consciousness, while retaining the expressivity of the original speech with natural breaks and co-expressive gestures. TypeTalker streamlines speech editing through a simple textbox that respects temporal alignment across edits. A comparative evaluation shows that TypeTalker reduces speech anxiety during live-recording, and offers easier and more effective speech editing facilities than the previous state-of-the-art interface technique. A follow-up study on recipient perceptions of the produced comments suggests that while TypeTalker's generic voice may be traded-off with a loss of personal touch, it can also enhance the clarity of speech by refining the original speech's speed and accent.

Author Keywords
Speech comments; multi-modal comment; automatic speech recognition; transcription error; self-consciousness; transcript-based speech editing.

ACM Classification Keywords
H.5.1 Multimedia Information Systems: Audio input/output; H.5.2. User Interface: Voice I/O; H.5.2 User Interfaces: Interaction styles; H.5.3 Group and Organization Interfaces: Collaborative computing.

INTRODUCTION
Speech commenting has empowered a variety of multimedia collaboration systems for online discussion [29, 55], writing/design feedback [26, 27, 45, 52–54], and cooperative storytelling [34]. Often in these systems, recorded speech serves as the core modality around which other interactive expressions (e.g., gesture, inking, and video) are weaved together in sync. The interface that enables the production and editing of speech comments is therefore central to the success of a multimodal collaboration system.

Recent studies report two major problems with existing speech commenting interfaces. First, speech commenters tend to be self-conscious during live-recording, distracted by speech disfluencies such as 'um' or 'uh', stutters, or long pauses [29, 39, 41, 55]. People may also feel disturbed when hearing their own voice [21, 55]. Second, while editing may be a step toward reducing self-consciousness, editing of spoken speech comments is taxing. Over the past decade, several transcription-based speech editing systems have shown that using a transcript as a proxy for audio offers effective semantic editing of spoken content [36, 41, 49]. However, these interfaces presumed an *a priori* audio transcription. To cope with the context of *spontaneous* speech commenting, these interfaces introduce separate modes of interaction: text-like audio copy and deletion, correcting live transcript from error-laden speech recognition, and re-recording for progressive revision [41, 49]. Tracking and switching between these multiple modes can easily confuse users [35].

To address these issues, we built TypeTalker, a speech synthesis-based multi-modal commenting system. In TypeTalker, the user's voice entry is transcribed to later be synthesized into a computationally refined generic voice. As we show, the synthesized voice reduces speaker anxiety, since the audio in the standardized voice lacks the linguistic glitches of the original speech, and doesn't cause the affective disturbance of hearing one's own voice [21, 29, 55]. In addition, we show that this approach reduces editing time by enabling an *error-laden* text transcription to act as a proxy for simultaneous editing of audio *and* any underlying metadata. This method allows temporal metadata, in this case gestures on a document, to be captured and replayed in sync with a speech comment, *even after the comment is edited.*

The functional benefit of TypeTalker's synthesis-based approach is its streamlined workflow for revising speech (Figure 1) in comparison with that of traditional transcription-based speech editing systems (Figure 2). While the captions (text) in a traditional interface only work as placeholders for utterances, TypeTalker guarantees a match

CSCW '17, February 25-March 01, 2017, Portland, OR, USA

ACM 978-1-4503-4335-0/17/03…$15.00
DOI: http://dx.doi.org/10.1145/2998181.2998260

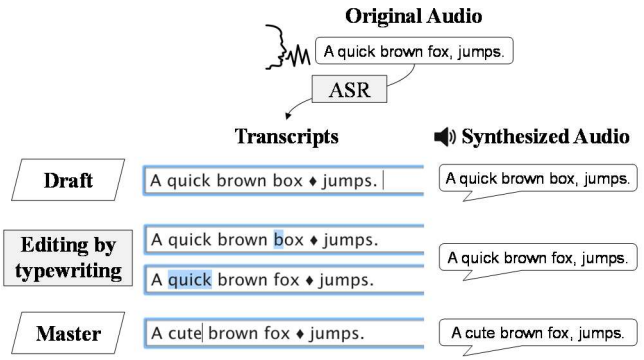

Figure 1. TypeTalker workflow. A user can finish different types of editing job in *one-pass*: correcting the ASR error ('fox' mistranscribed as 'box'), and changing a spoken word content (from 'quick' to 'cute').

between audio and text because the voice is synthesized from the text. This approach enables simpler and more efficient revision, as caption correction and content editing are unified through *single-mode keyboard editing* over the transcribed text. For example, TypeTalker supports generative editing operations (e.g., insertion of new words or syllables) via simple text editing, while the traditional approach requires recording (or re-recording) additional voice content.

To evaluate the efficacy of TypeTalker's design, we conducted two comparative evaluations against a state-of-the-art peer system built under the traditional approach [41]. The primary study asked participants to produce voice comments in each interface, and found that TypeTalker's use of a synthesized voice significantly reduces public/social self-consciousness. Comment producers also unanimously reported that TypeTalker's streamlined workflow enabled more lightweight and efficient speech editing compared to the previous technique. A follow-up study let participants listen to two types of voice comments generated from the primary evaluation. While some participants reported that they missed the personal touch of human voice when listening to TypeTalker's machine-generated voice, some also felt the machine was easier to understand compared to the human condition, thanks to its clear articulation and steady pace. Together these two studies highlight the potential and possible trade-offs of our approach in a constructive and collaborative use context.

RELATED WORK

Our study was inspired by previous work on speech-centered multi-modal collaboration systems and on psychological analysis of user behavior when using such systems.

In the field of computer supported cooperative work (CSCW), speech has been used as a central mode of communication in many different multimedia collaboration systems for supporting discussion [29, 55], feedback [26, 27, 45, 52–54], and storytelling [34]. Jonathan Grudin pioneered studies on issues surrounding voice application in a cooperative setting [17]. His study was first to raise

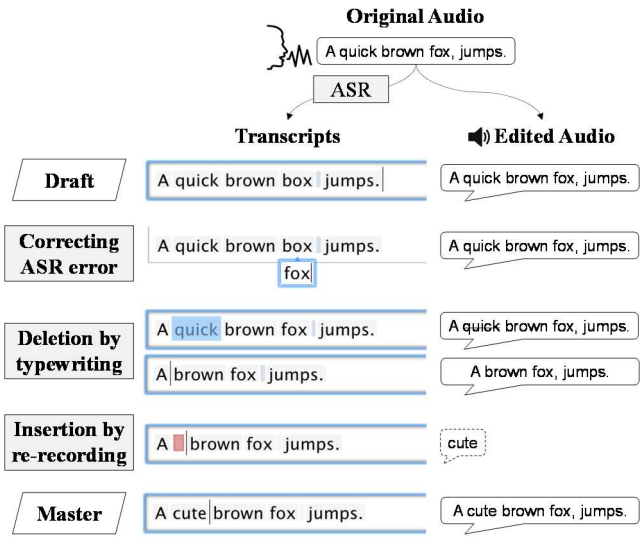

Figure 2. The traditional transcription-based speech editing workflow requires a user to switch between three different input modes: correcting captions ('box' to 'fox'), editing contents (deleting 'quick'), and re-recording (adding 'cute').

awareness about the speech *consumption* problem: that browsing often takes longer than speaking. The subsequent studies of the field solved that problem by introducing interactive systems using binary acoustic structuring [20] and waveform indexing [55]. In this study, we aim to tackle the *production* side of problem, which has been frequently documented [19, 29, 39, 41, 55] but not yet resolved.

Use of rich media allows communicators to express equivocal and nuanced ideas more effectively by clarifying ambiguities [13]. For instance, voice comments have proven more effective than textual comments for writing feedback as they convey nuances and emotions of the commentator [10, 29, 55]. In addition to speech, researchers have combined various types of visual cues to enrich spoken comments. Ink mark-ups act as a visual mnemonic or future retrieval for indexing [26, 52, 53]. Gesture provides a deictic cue that delivers indexical information that is co-expressive to speech [45, 54]. Moreover, video can leverage embodied nuances such as facial expressions and upper-body gestures to enhance communication [14, 25]. Our interface also benefits from co-expressive speech + gesture commenting, but it goes beyond previous work by resolving the problems of speech recording in terms of user psychology and ease of revision.

Speech comment editing is a well-explored topic in HCI. Through the years, different designs to speech editing interfaces have been proposed and evolved. In professional audio/video production software, low-level audio editing was made possible through waveform representation over a timeline [2–4, 6]. Since the fine-grained timeline in the waveform interface introduced an extra burden on novice users, especially in the context of speech editing, researchers structured audio into a higher-level representation by

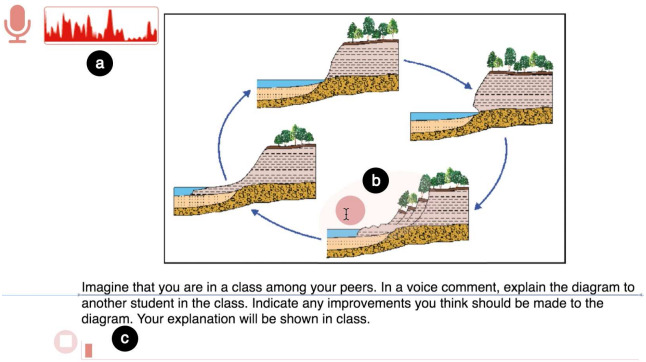

Figure 3. TypeTalker inside the RichReview system [54] is designed to record, edit, and replay speech + gesture comments.

analyzing its acoustic structure, such as speech/non-speech chunks, phrases, or sentences, for browsing [5, 44] or editing [1, 20]. A more recent approach employs time-synchronized captions from automatic speech recognition to augment the edited audio chunk with semantic meaning. This approach enables visual skimming and browsing of audio without requiring users to spend time tediously re-listening [9, 38, 47, 51]. Moreover, snapping the editing prompt to a word's boundary afforded text-like audio/video editing [7, 36, 41, 49, 54]. Contrary to the previous transcription-based systems where audio is loosely coupled with the transcription, our system's synthesized audio is guaranteed to say exactly what is written and edited as text. This approach not only enables faster speech revision, but also supports generative editing operations, such as insertion or rephrasing of words that used to require cumbersome re-recording, through keyboard input.

Captions from ASR have transcription errors by nature. On one hand, words from the error-laden transcript have been shown to act as semantic cues for extracting key points or as navigational cues for browsing [51, 54]. On the other hand, recognition errors hurt listener comprehension [43] especially for non-native speakers [33]. When the recognition rate is very low, listener comprehension might be as bad as [47] or even worse than [31] that with no transcription at all (e.g., reading homophone transcription errors can induce confusion by misleading listening comprehension [54]). While all of these studies suggest the impact of transcription errors on *recipient* psychology, a recent report from Sivaraman et al. suggests that the pressure to correct transcription errors may also increase the workload of speech *producers* [41].

Previous research on speech interfaces found design factors that affect speaker psychology. While ordinary voice conversation is ephemeral, if speech interfaces give people a sense of being recorded, they tend to speak differently [11]. Even a physical awareness about the recording device (e.g. awareness about a wired lapel mic) can impact speaker response, reducing creativity and disclosure and introducing speech disfluencies [48]. Recent studies on asynchronous speech commenting systems suggest several factors that

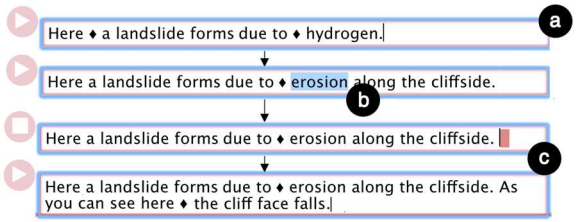

Figure 4. Editing process for the spoken comment

might increase cognitive load for speech commenting: lack of lightweight editing features, concerns about one's speech disfluencies, and affective disturbance of hearing one's own voice [21, 29, 55]. Our qualitative results suggest how the use of voice anonymization and ease of revision can reduce speaker anxiety in speech editing interfaces.

DESIGNING TYPETALKER

The main design contribution of TypeTalker is a novel interface for editing speech + gesture comments as text in a textbox. To start, we provide the reader with a brief summary of the RichReview system on which our TypeTalker design and implementation is based.

Building TypeTalker Based on RichReview

The TypeTalker editing interface was integrated and evaluated in the context of the existing multi-modal collaborative annotation system RichReview by Yoon et al. [54, 55] (Figure 3). RichReview enables speech + gesture commenting on PDF documents. To record a comment with RichReview, the user highlights a line of the document and 'tears' a space beneath The system offers time-synchronized recording of mouse gestures that can effectively augment verbal description of speech with a deictic visual aid of an animated blob (e.g., Saying "*This* section needs a revision." while dragging over text, Figure 3. (b)). These gestures are time-aligned with the recorded speech and replayed in sync with it. Users can replay their comment by pressing the 'Play' icon and navigate during replay by either (1) clicking on passages of text or (2) clicking a ghosted remnant of an associated gesture.

TypeTalker Design Rationale

We set the following three objectives to make the design decisions which embody the needs and challenges learned from prior work.

Reducing self-consciousness A sense of being recorded can introduce anxieties in speech production. Two major sources of anxiety are the speaker's concerns about the way one's voice will sound to the recipients (e.g. 'um's), and the affective disturbance of hearing one's own voice. By using an automatically generated text-to-speech voice instead of the speaker's own voice, we let the user be less self-conscious about recording their voice.

Single-mode speech editing Previous editing systems [36, 50, 54] maintain a loose correspondence between text token and *source* audio snippets. This setting (1) requires producers to do "double-work" for editing audio and

correcting transcription errors, (2) introduces confusing interaction modes between audio editing and caption editing, and (3) slows down frequent and small edits (e.g., adding the past-tense ending, "ed"), since new words can only be inserted by speaking. In TypeTalker, synthesized audio is generated based on text tokens as the user edits them. This tight coupling of edited audio to text tokens enables a unified single-mode revision process for audio (Figure 1).

Retaining expressivity of the original speech + gesture recording A pure synthesized voice generated based only on the transcribed text misses the richness of multi-modal inputs, such as pauses in the user's speech or co-expressive gestures. In TypeTalker, the synthesized speech retains expressivity of the source speech, such as natural pauses. Also, time-synchronized gestures recorded with the original speech are transferred to the corresponding words of the synthesized voice.

Editing Interface

TypeTalker allows editing of a speech + gesture comment through standard keyboard-based text editing. In this section, we illustrate a typical user workflow.

Imagine that a user wants to comment on a diagram. In our example, she begins a new comment beneath the prompt. A tooltip waveform and icon blink to remind the user that they are in recording mode (Figure 3. (a)). While recording, she can refer to areas of the diagram by making deictic gestures (see (b)). Inside the text box, a red marker pulses at the place of insertion, reminding the user that they are in recording mode (c).

Once the user stops recording, the system is ready to support editing (Figure 4). It first swaps the blinking marker with the final ASR transcription results (a). In TypeTalker, each word of the ASR transcript is linked to time-stamped metadata (such as a gestures). To enable text-like single-mode editing, the system presents the transcription as a normal text inside the textbox by managing this audio correspondence data in the background, hidden to the user.

At this point, the user can review their comment and edit it through standard keyboard-based text editing. It is important to note that one can both *correct transcription errors* and *insert new content* in the same textbox with a seamless and consistent keyboard interaction. For instance, in (b), the user fixed the mis-transcribed 'hydrogen' to 'erosion' as well as typed-in "along the cliffside"). Editing can also include deletion of portions of speech, punctuation revision, and pause manipulation. When the user wants to add new contents with gesture or speech pauses, they place their cursor at the end of the text and press the 'Enter' key to begin a new recording (c). The same revision process follows.

Upon pressing the 'Play' icon for playback, the system narrates the newly edited text, together with an animated visualization of gesture and pen input properly synchronized. This multi-modal replay gives a vivid multi-modal rendering that supersedes just reading the transcribed text.

One of our goals was to retain some of the expressive quality of the user's original voice. Of particular concern was the fact that speech-to-text lacks the user's natural breaks in speech. We alleviated these concerns by transferring pause from the original speech to the synthetic voice. To help users control which pause they would like to keep, the in-line markers '♦' denote a short pauses, while we append a period '.' for longer pauses. We initially also transferred select pitch contours per-word according to their root-mean-squared-error with the synthesized word's contour; however, we were not able to find the right balance between the articulated prosody of speech-to-text voice and the user's natural prosody. The system used in our evaluation had the pause transfer feature only, but more than half of the participants reported that the retained pause timing could effectively convey the majority of expressive richness in the original speech even without having prosody transfer.

ALGORITHMIC METHODS

In this section, we describe our implementation methods in roughly chronological order from recording to playback.

Real-time Transcription

For transcription, our system streams microphone data to the IBM Watson service [22], which we also use for synthesis. For each recognized word we obtain timestamp data. As the system revises its matches, we store the current best match, and permanently append the final matches at the current insertion point when the user stops recording. Any special markers received from Watson are ignored.

Our early prototype opted to show the live transcription results in the textbox while recording; however, we observed in the first pilot study that on-the-fly transcription errors distracted users from their commenting job, because they became concerned with spotting and fixing errors as the transcript updated. Therefore, we opted to minimize distraction by presenting a simple blinking marker only, much like a text-editing caret.

Transcript Post-processing

The ASR transcript is bereft of textual formatting like capitalization and punctuation. Since formatted text reduces workload (users don't have to manually punctuate) and eases comprehension of the text, we implemented pause-length-based heuristics for automatic punctuation and capitalization of the transcript.

Inline pause markers retain some of the original richness of the user's voice when synthesizing voice from the transcript. Although our system cannot detect grammatical commas, we can transfer their audible effect (a break in the sentence) in a homogenized fashion with other punctuation, such as hyphens. We chose to show pauses to users rather than hiding them to allow users to remove unnatural breaks in their speech, for instance due to hesitation. For both interfaces in our evaluation, we analyzed the final match for pauses by marking gaps between timestamps. We inserted a pause marker between words of the transcript if the gap is greater

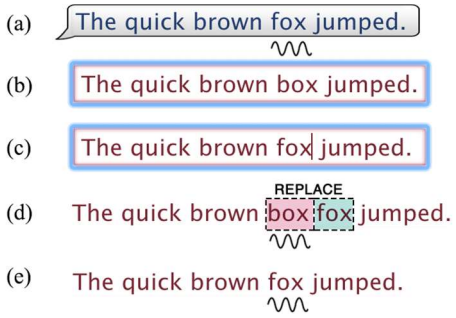

Figure 6. (a) The user speaks and gestures. (b) The user finishes recording, and ASR results are inserted into the textbox. (c) The user changes "box" to "fox." (d) The edit detected as a replace operation. (e) The edits are compiled into a final version. Notice that the gesture remains.

than 30 msec. For TypeTalker, pause markers are a '♦' symbol (for gaps less than 1 sec) or a period (for gaps 1 sec or greater). Our use of pause markers instead of commas is similar to the pause tags found in [36]. We automatically capitalize a word following a period.

During voice insertion, we also capitalized the first word if the previous word ended in a period. This minimal punctuation was introduced to shorten user editing time and make voice insertion fluid and intuitive.

Realignment and Synthesis

Our system incorporates gestures as extra-modal temporal metadata linked to speech recordings. Since the user can freely edit text, any edits to the transcript must be reflected in the playback of these gestures. For instance, if the user deletes a sentence, any temporal metadata recorded "inside" that sentence should also disappear. We opted to realign gestures by comparing the edited text to successive versions of previous text, starting with the original transcript.

Figure 6 illustrates our approach through a simple example. In (a), a user records their voice and gestures while speaking the word "fox." The machine transcribes "fox" erroneously as "box" (b). When a new transcript is inserted from voice recording, the text's current state is saved as a sequence of *tokens*. A *token* links a transcribed word (as text) to the segment of recorded audio in which it appeared (as timestamp info). We call this initial token sequence the "base" tokens. Pauses both before and after a base token are stored so that the length of each pause marker on playback can be deduced by the tokens nearest to it. Gesture timestamps and other metadata is stored separately.

Next, the user corrects the mis-transcription using her keyboard. When she presses Play (c) the comment is compiled. Compilation comprises three steps:

(1) stripping both the base text and the edited text of any punctuation and pause markers, as well as making the text lowercase
(2) comparing the two texts with a string edit difference algorithm [32] to compute a sequence of operations on the base text. The four operations are: insert (insert a token before the current token at the index position), delete (remove the token at the index position), replace (change the token's word at the index position), and unchanged (do nothing). The replace operation also stores the edited word. The difference algorithm infers which words are 'inserted' or 'deleted.' The replace operation is a concatenation of back-to-back 'delete' and 'insert' operations (d).
(3) applying these operations to the base tokens. For a delete operation, the token is deleted. For an insert operation, a new token is inserted at that point in the sequence. This token has no timestamp information, as it was generated through typing. For a replace operation, the word stored in the base token is replaced by the edited word. The token's timestamp information remains unaffected. In addition, pause information for output tokens is computed by comparing the remaining pause markers in the edited text with the base tokens.

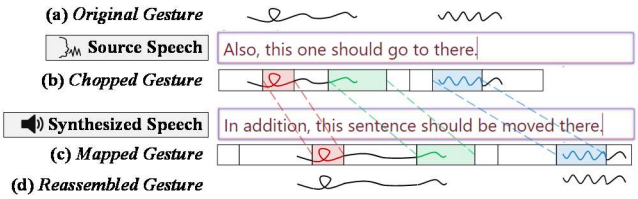

Figure 5. The split-map-reassemble process for gesture transfer.

The output is a new "edited" token sequence. For playback, these edited tokens are flattened into a transcript and converted to Speech Synthesis Markup Language with <break> tags for pauses. TypeTalker also dampens user pauses along a sigmoid curve that approaches our synthesizer's length for pauses after a period (~450ms).

Mapping Gestures to the Edited Tokens

In TypeTalker, gestures recorded during speech must be automatically remapped when the user edits the text. Since the synthesized voice is spoken at a different rate than the user's, gestures made during recording also need to be stretched. Both features rely upon situating words in the synthesized audio. Since timestamp information was not available for the speech-to-text output, we ran the audio and edited transcript through HTK forced alignment [8] to obtain timestamps. From both the edited tokens and the speech-to-text timestamps, we then computed a sequence of "synthesized" tokens. Gestures could then be recomputed from *both* the edited tokens and the synthesized tokens through a split-map-reassemble process similar to Golovchinsky and Denoue's visual segmentation scheme [16]. Our method respects the time correspondence of speech tokens and gesture strokes. First, in the splitting phase, gesture strokes in the original recording (Figure 5. (a)) are chopped into pieces of strokes (see (b)), where temporal information corresponds with co-occurring tokens in the edited sequence. The chopped pieces are then mapped to the

corresponding speech tokens in the synthesized sequence (c). Finally, we combine the reassembled gesture-piece sequence with potentially clipped pieces by lumping consecutive runs of gesture pieces into a single continuous gesture stroke (d) that respects the new beginning and ending time-stamps.

Mapping across Multiple Edits

Since it is unreasonable to expect users to make their comment in a single pass, our system supports insertion of speech + gesture inside existing comments. This raises the technical question of where the new recording's tokens should be inserted into the previous base sequence, since the user may have made several textual edits, maybe even deleting entire sentences. In our system, when the user inserts a new voice comment in-line, we set the base tokens to be the previous edited tokens, and inject the recorded tokens at the insertion point. In other words, the previous edits to the text are "committed" to be the reference for future edits. We found this approach effective at maintaining gestures across multiple edits, even when the user strays far from their original comment.

Performance across Multiple Edits

Rather than sending the entire comment to speech-to-text on each replay, only sufficiently changed sequences of text are synthesized. The previous edited token sequence is chunked by punctuation and pause duration, and each section is compared to the new edited sequence. The synthesis result(s) are then stitched together with any preexisting sections of audio for the final comment.

Implementation

TypeTalker was implemented in the browser with Javascript, and utilized a back-end Python server for interfacing with HTK and other off-the-shelf audio analysis tools. In our prototype, WAV files were sent locally to the server for analysis. Since our synthesizer returned audio the quickest when the ogg-opus codec was specified, we also wrote a client-side decoder to convert ogg-opus to standard WAV format to improve synthesis speed.

PRODUCER-SIDE EVALUATION

Our primary evaluation aimed to study whether our new design approach of TypeTalker could reduce *producer* speech anxiety and promote faster speech editing by comparing it to the SimpleSpeech system [36], a design based on the traditional approach. To draw out a quantitative comparison as well as qualitative implications for the future design improvements, we employed quantitative-major embedded design mixed methods where a task-driven lab study embeds exploratory qualitative inquiries such as observation notes and interviews [12].

Peer System

Our interface was evaluated against a prior speech editing interface, SimpleSpeech [41], that adopts the previous approach. SimpleSpeech presents speech audio as a series of word tokens that affords text-like editing operations, including deletion, copy, cut, and paste. Transcription correction can be done in a small pop-up box that appears when the user first selects the target word token and then types substitute texts. As shown in Figure 2, it also has confidence shading [47], instant word replay, and a feature for running through the words by token. Other than those core features all the interactions remain consistent across the two interfaces.

Sampling

For this formative evaluation process, we recruited 15 young (18-22 years old, 14 female) undergraduate students at a US university. We selectively sampled participants who speak native or fluent English (12 native speakers), since the speech recognition system was optimized for standard American English pronunciations and accents. Our participants had different majors spanning across art, science, and the humanities.

Data Collection

To set up a concrete and substantive use context, we put participants in the shoes of a student who takes part in the discussion activities of online coursework at a University. More specifically, we gave participants a series of commenting tasks that asked them to record their speech and gesture on given diagrams. The diagrams depict middle-school level academic topics, such as the 'bottle recycle process' or the 'coastal erosion process' shown in Figure 3. We only selected diagrams with very easy concepts and minimal text, because we wanted the participants to focus on our interface rather than spending too much effort thinking about what to say. Each participant performed 3 sessions of tasks; in each, they created a paragraph-long speech comment. These tasks imposed proper amounts of effort on the participant to the extent that they had to leverage the full functionalities of the system in a reasonable time range (total 3~6 min) for this 1.5 hour-long study.

After the sessions for each condition, participants answered a set of surveys for rating perceived public/private speech anxiety and overall task loads. The anxiety measure was adopted from the Scheier & Carver's Self-Consciousness Scale (SCS-R) by selectively contextualizing four questions about public speech to the asynchronous speech recording use case [37]. The workloads were measured as the weighted NASA-TLX scale [18]. Participant activities were also logged to measure the number of different recording and editing behaviors as well as the transcription results.

To collect the qualitative data, the investigator sat behind the participants' workbench, observed her task practices, and took notes of any notable incidents. Implications from the observations were referred back from the post-task interview for two purposes. First, we asked 'how' and 'why' questions to the participants to better understand the rationale behind their behaviors [28]. Second, we did member checking [30] to validate our on-the-fly interpretation of participant behavior.

The task sessions were followed by a ~15 min-long semi-structured in-depth interview. The interview was structured

in a top-down manner where the general implications were asked first, and more specific leading questions followed. The topic of the questions covered usability & performance, learnability, speech anxiety, editing behaviors, strategies for managing speech recognition errors, comparison of speech commenting to face-to-face speech or textual commenting, etc. Also, the interviewer sometimes brought up odd behaviors observed in the interviewee's task sessions to better understand what happened and how they felt about it. Entire interview sessions were audio-recorded for potential future analysis.

Data Analysis

We performed the paired t-test for the quantitative data, such as the self-consciousness or workload indices. For generating quantitative implications, we conducted theoretical sampling [15] by comparatively analyzing data from the two different UIs. After collecting and transcribing interview data into texts, the lead investigator performed an open-coding followed by a flat-coding to draw out theoretical categories of the implications in consultation with the coauthors. To maximize the validity of our findings, we triangulated different types of data, and consistently looked for negative cases to falsify potentially defective evidence [30].

RESULTS

The participants generated a total of 90 comments (15 × 2 × 3) for the tasks. On average, the comments were 20.3 sec long (SD = 16.1) with 39.6 words (SD = 36.2) for TypeTalker, and 18.0 sec long (SD = 14.3) with 36.9 words (SD = 35.4) for SimpleSpeech. They also made a couple of gesture strokes for each session (M = 1.50, SD = 2.08). None of these basic measures were significantly different between the two conditions.

Average recognition accuracy of the source speech measured as word error rate (WER) was .19 (SD = .10) in the TypeTalker condition, and .16 (SD = .09) for the SimpleSpeech condition which are slightly higher than IBM's official data of .104 [42] possibly due to the participants' speech disfluencies. The response speed of the transcription engine was near real-time as we live-streamed the audio to the Watson's cloud server in the 16-bit 22.05 kHz PCM format.

Reduced Self-consciousness

The participants perceived significantly less public/social self-consciousness during speech when using TypeTalker, thanks to the synthesized generic voice imposed less concerns about public performance than the peer system that records audio as is (see Figure 7, left). In our SCS-R measure, ratings for public/social-anxiety were significantly lower with TypeTalker (M = 2.57, SD = 1.13) than SimpleSpeech (M = 3.06, SD = 1.05, p = .019, paired t-test, Cohen's d = .46).

From the qualitative responses, we found that a total 12 of 15 participants reported lowered self-consciousness. First, 7 participants (P1, 2, 4, 8, 10, 13, and 14) reported that the

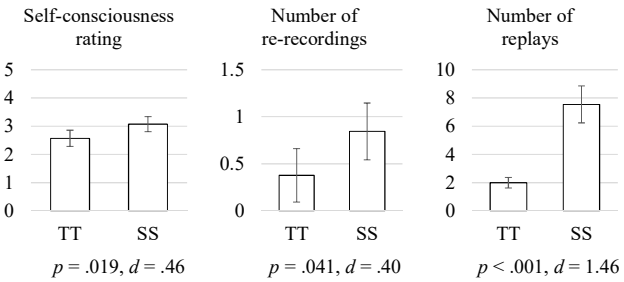

Figure 7. Quantitative results from TypeTalker (TT) and SimpleSpeech (SS) conditions (95% confidence intervals).

TypeTalker interface alleviated their concerns about the way their speech will sound to the recipient, because the machine's voice doesn't retain small speech disfluences including 'uh', 'um', stutters, hesitations, or long pauses. In contrast, voice recordings in SimpleSpeech made them "nervous that I would just keep going like 'um, um...' in the middle of my statements. It just seems like there was a lot more that could go wrong that way. (P13)". There was *no* participant in our sample who felt more anxiety in the TypeTalker condition.

Second, 7 participants (P2, 4, 5, 6, 7, 11, and 13) liked that they don't have to listen to their own voice, which often causes affective disturbance [21]. To quote P13, "I personally hate listening to my own voice on recordings [laugh]. It's weird to me. It's a little off-putting (P13)." This disturbance remained salient during the revision phase, as P11 stated "I don't like hearing my own voice. So when I try to replay them, I almost muted the computer". The other 8 participants didn't mention the affective disturbance from hearing their own voice.

Finally, 6 participants (P4, 5, 8, 9, 13, and 14) were less concerned about making mistakes while using TypeTalker because they "knew that it was easier to correct those mistakes (F5)" and "required much less work (F8)" using the keyboard interface afterwards.

Effective Revision

The participants were unanimous that TypeTalker's type-written editing was not only easier to learn, but also more lightweight and effective for editing. The participants liked familiarity of the normal text editing interface, single-mode, and no need for re-recording. This quote summarizes the implications well:

> *"TypeTalker was easier, just because it was very similar to like normal typing, I could just go in and fix things, and you know I could change words, I could change sentences if I wanted to without having to worry about it, whereas with SS, if I wanted it to change or rephrase something, I have to go in and re-speak it, and it usually comes out sounding different than like louder, like just awkward (P14)."*

Reduced confusion from unified editing mode

A total 9 of 15 participants said that TypeTalker's single-mode editing was straightforward to learn and use, and more efficient. Although most of them felt that SimpleSpeech's interface was easy enough to get accustomed to in a reasonable learning time, they oftentimes felt the interface "confusing (P6)," "frustrating (P8)," or "not knowing what to press (P5)". They felt that there were too many options, and as P12 stated, "when to say, when to type, when to press enter was different than expected". This suggests that TypeTalker's design decision to unify audio and text editing significantly improves editing efficacy. For some users, audio-text division of SimpleSpeech was not only about the mode confusion issue during editing. Sometimes, when users accidentally deleted an audio token, they tried to recover it by typing that transcription to the nearest token, losing audio although showing the correct transcript: "I often deleted my voice recording closing a glitch. That was the hassle (P10)."

Efficient content editing

Editing spoken content was more efficient in TypeTalker, because, unlike in SimpleSpeech, it didn't require participants to re-record parts (11 of the 15 participants). When asked about how hard they worked for editing content other than transcription errors, the survey response showed a trend that they edited more content in TypeTalker than SimpleSpeech ($p = .082$, Cohen's $d = .47$). More pauses were edited during the revision process as well. In the original speech data, the number of pauses were not significantly different, but the end results had significantly less pauses in TypeTalker condition. This is possible because of a trend where more pauses were deleted during editing in the TypeTalker system ($p = .093$, Cohen's $d = .33$).

To insert new audio contents in SimpleSpeech, users had to re-record the part of speech, since there is no way to create new audio from the edited text. For our participants, this re-recording worked as the major drawback for editing content (11 of 15 participants). This implication is reflected in log data that shows that users rarely re-recorded in the middle of a SimpleSpeech stream (M = .84, SD = 1.19). Also, when they were given the typing capability to add contents in TypeTalker, the use of the insertion feature became significantly rarer (M = .37, SD = 1.12, $p = .041$, Cohen's $d = .40$), because they preferred to just type to insert voice rather than recording that part again. They preferred not to re-record speech not only because it was cumbersome, but also because the inserted voice sounds awkward and felt "forced in (P5)", "misplaced (P8)", "louder (P14)", "off-flow (P14)", or "choppy (P15)".

Implications on correcting transcription errors

Even though there seems to be more pressure to correct transcription errors in TypeTalker than SimpleSpeech "because the program will specifically read what was transcribed (P8)", such pressure was evened out by the three factors beneficial to the TypeTalker condition. First, editing by keyboard input was easier in TypeTalker as stated above. Second, SimpleSpeech users also felt pressure to fix mis-transcriptions, so that the recipient of the messages wouldn't be confused by the wrong text (13 of the 15 participants). Finally, editing transcriptions in SimpleSpeech forced the participants to re-listen to their audio, because they had to *match* the text to the voice. There were significantly more replays in SimpleSpeech than TypeTalker ($p < .001$, Cohen's $d = 1.46$). Re-listening was upsetting for the participant, not only because it was cumbersome (P2, 7, 8, and 14), but because re-listening during editing (P11) also caused the self-disturbance problem of hearing one's own voice.

Nonetheless, some other users liked that they could re-listen to their voice in SimpleSpeech, because it helped remind them of the content of their original narration (F9 and 11). Although the TypeTalker system didn't have the re-listening feature for replaying the original voice, one might improve the transcription correction process by including an in-situ replay feature as a mnemonic device for the system in the future (e.g., replay the snippet of original speech, when selecting words for editing).

Valued Richness of Original Audio

8 of the 15 participants liked TypeTalker's pause mark feature that transfers subtle timings from the original voice to the machine's voice. The pauses in the machine generated voice could make it "sound more human-like (F5)", and enabled them to verbally "emphasize (F2)" a phrase by generating some temporal suspense. Also, listing items such as "Croatia <pause>, Slovenia <pause>, and all (F2)" sounds more natural with having the pauses in-between. This implies that future machine-synthesized voice technologies can largely benefit from transferring richness of the original voice to the synthesized voice.

Although all producers admitted that the machine's voice reduces speech anxiety and enables efficient editing, 8 missed rich acoustic expressions from their own voice, such as "emotion (P8)" or nuances (e.g., "sarcasm (P9)"), delivered by subtle "inflections (P4, 6, 14)". A few (P3, 4, 14) also wanted to retain the identity of the original speaker (e.g., "gender (P4)"). Producers may have been concerned that this loss of expression would impact the recipients of their comments. To explore how comment *consumers* were affected by the machine voice – whether they, too, missed natural expression, and to what extent – we conducted a follow-up qualitative study, described in the next section.

CONSUMER-SIDE EVALUATION

The goal of this follow-up study was to understand how the content-consumer's comprehension and experience are influenced by the two types of voice comments generated from each interface: TypeTalker with a machine's voice, and SimpleSpeech with a human voice. For this study, we collected qualitative data from participants who conducted a set of consumption tasks on the comments produced during the first study.

Sampling

We recruited 10 (19-39 years old, 6 female) participants at our university. We diversified the consumer demographics by recruiting participants from varied academic backgrounds. Also, unlike the primary evaluation, 4 were non-native English speakers comfortable in written English. None of them had participated in the producer-side evaluation.

Tasks

To mirror the task context of the primary study, we placed participants in the shoes of a student in an online peer discussion context, and asked them to critique producer-generated explanations of various diagrams, focusing on audio delivery. Specifically, we let them first *listen* to each speech comment, and then *type* a short response (2-4 sentences) evaluating each of them. We explicitly asked them to play the audio rather than reading the transcribed texts, so that they could listen to the comments in order to compare generic and human voices.

Procedure and Materials

At the beginning of the study, the investigator gave a brief tutorial about how to use the interface to create a text response, then began the first session. There were a total of 2 sessions, one for each interface condition (TypeTalker and SimpleSpeech) which lasted a total of ~45 minutes. In each session, the participants conducted 2 commenting tasks that took ~5 min each. Each task contained a comment randomly assigned from a producer in that condition from the primary study, with the constraint that no diagram was presented to each participant more than once. Condition order was counter balanced. After both sessions, the study was concluded with an audio recorded, ~10 min-long semi-structured interview. The full study took 1 hour.

Results

Consumers were ambivalent as to which voice type they preferred. 5 (C2, 5, 6, 7, 9) preferred a human voice in general, but were also not particularly bothered by the machine voice. 4 (C1, 4, 8, 10) did not express a clear preference for either voice, and one, a non-native speaker (C3), preferred the machine voice. Even those consumers who preferred the human voice did not find that the machine voice hampered their comprehension. For instance, C2, who preferred the human voice, stated, "The voice, although robotic, was concise and it was relatively easy to follow along with the diagram." C5, who also preferred the human voice, said that "the machine voice itself didn't bother me. It was fine." 2 participants (C1, C10) did not even notice there was a difference between conditions until pressed.

In general, consumers cited improved elocution through standardization as a major benefit of the computer generated voice, with possible trade-offs of lost expressivity and engagement. For example, C2 thought the machine voice was "easier to follow because it's easier to understand a slow robotic voice," while C8 found the machine voice preferable "if someone has an accent, or speaks really fast or slow," and remarked that a standardized voice "would be helpful for a wider range of students." Awkward lengthy pauses, disfluencies, and speaking rates were continually cited as an issue for human-voice comments. C3 explained, "it's much more comfortable to listen to the machine voice for me. Because the human voice, they have pauses and they [speak] more slowly." Even C9, who preferred a human voice in general, admitted, "the [human voices] had a lot of awkward pauses. That does follow the natural way of speaking, but because of that, it also was more difficult and unclear. There's a lot of rapid changes in pace. So it's like a pro and con."

Nevertheless, some consumers felt that the benefit of a standardized voice also imposed an effect on their engagement. C9 went on to state, "when I am listening to a machine, it is a little harder to engage […] Because it's like a one-tone voice, and one-speed." In addition, C7 found a human voice more "soothing" and "easier to pay attention to" than the "monotone" machine voice. Future work to transfer prosodic features could remediate or remove this drawback.

TypeTalker's improved text output was also appreciated. 9 out of 10 participants found the text useful to their comprehension of the comment, especially for reminding them of the audio (the last did not mention the text). In the human condition, 4 participants cited issues with SimpleSpeech-produced comments: two (C6, 10) noticed typos and were "a bit confused (C6)" as to the mismatch between speech and text, while the other two (C4, 8) complained about punctuation and grammar. However, two participants (C5, C1) in the TypeTalker condition also mentioned correcting minor mis-transcriptions in their remarks to the commenter, which highlights the addition burden placed on commenters by ASR accuracy. Many users could not comment directly on the quality of the text as they found the quality of speech enough for their understanding.

DISCUSSION

From the findings of our evaluations, we gleamed several implications for the design of future speech editing interfaces.

Implications from the trade-offs between human and standardized voices

Most of participants in our producer study felt that the synthesized voice sounded more professional whereas the human voice has a better personal touch. They thought that the professionalism of TypeTalker comments would be better received in formal or official settings (e.g., lecture, audio book publication, etc.), while for other settings, such as Snapchatting or a lecture in a smaller class, use of their own voice would fit better, since it can convey character and personality of the speaker. Interestingly, participants in our consumer-side study made similar remarks. This implies that the *use-case* (or work context) of the speech commenting system might be one of the deciding factors on which way to present spoken comments.

Results from our consumer-side study suggest that standardized voice might enhance the listener's comprehension by reducing distracting aspects of speech such as disfluencies, lengthy natural pauses, and fast speaking rates. This effect was particularly noticeable for non-native English speakers, since for them, comprehension of speech was a priority. This implies that TypeTalker may enable a more diverse group of individuals to hold a discussion than that accomplished by recorded voices alone. This would be especially useful for a multi-cultural CSCW context. However, more work needs to be done to improve the naturalness and expressivity of the machine voice, in order to tackle the trade-off from the loss of personal touch.

Retaining more personal touch from the original speech

Even though the transferred pause timings could convey temporal subtleties of speech, such as rhythm and suspension for emphasis, other acoustic qualities remained lost. For example, transferring natural intonation, prosody, and loudness of the original speech could make the synthesized voice sound more similar to the original. The speech synthesis research community has been presenting a set of technologies, such has pitch-synchronization [46] or emotional prosody modelling [40], that could be used to realize such features.

Hybrid approach: mixing original and synthesized voice

Future designs could take a hybrid approach that takes different advantages from both of the approaches by acoustically mixing the synthesized voice with the original speech. This could enable type-written generation of audio without having to re-record that part of speech. The latest speech conversion technique promises seamless stitching of synthesized voice into an existing speech stream [23]. Also, a speaker de-identification technique can be used when users want to anonymize their voice for reduced self-consciousness [24].

CONCLUSION AND FUTURE WORKS

This paper presents TypeTalker, a multi-modal commenting system that helps reduce self-consciousness of live voice recording by substituting a user's voice for a synthesized one while respecting the temporal alignment of extra-modal metadata such as marks and gestures. Our system's keyboard based interface supports a variety of revision needs, such as correcting transcription errors, deleting existing words, and inserting new words, under a consistent and seamless single-mode workflow. We evaluated efficacy of TypeTalker in comparison with SimpleSpeech [41], one of the latest speech commenting interfaces. Results from the producer-side study show reduced speech anxiety and more efficient revision when using TypeTalker, while results from the consumer-side study suggest the promise of enabling more types of speakers to communicate effectively.

Finally, our producer evaluation assumed moderately high ASR accuracy in a quiet lab setting. In reality, multiple factors can degrade the transcription quality: environmental noises, unavailability of high quality transcription services, or non-native speakers. This could have an impact on both editing time and speech comment consumption. It would be interesting to explore how the dynamics of design and user experience change in scenarios where transcription error rates are high.

ACKNOWLEDGEMENTS

Yoon gratefully acknowledges support from the Kwanjeong Educational Foundation. This work was supported in part by gifts from Microsoft.

REFERENCES

1. Stephen Ades and Daniel C Swinehart. 1986. *Voice annotation and editing in a workstation environment*. XEROX Corporation, Palo Alto Research Center.
2. Adobe. 2016. Premiere.
3. Adobe. 2016. Audition.
4. Apple. 2016. Garage Band.
5. Barry Arons. 1993. SpeechSkimmer: Interactively Skimming Recorded Speech. *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, ACM, 187–196.
6. Audacity Team. 2016. Audacity.
7. Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2012. Tools for placing cuts and transitions in interview video. *ACM Transactions on Graphics* 31, 4: 1–8.
8. University of Cambridge. HTK.
9. Juan Casares, A Chris Long, Brad A Myers, et al. 2002. Simplifying video editing using metadata. *Proceedings of the conference on Designing interactive systems processes practices methods and techniques DIS 02*: 157–166.
10. Barbara L. Chalfonte, Robert S. Fish, and Robert E. Kraut. 1991. Expressive richness: A COMPARISON OF SPEECH AND TEXT AS MEDIA FOR REVISION. *Proceedings of the SIGCHI conference on Human factors in computing systems Reaching through technology - CHI '91*, ACM Press, 21–26.
11. Herbert H Clark. 1996. *Using Language*.
12. By John W Creswell, Vicki L Piano, and Clark Published. 2007. Designing and Conducting Mixed Methods Research. *Australian and New Zealand Journal of Public Health* 31, 4: 388–388.
13. Richard L. Daft and Robert H. Lengel. 1986. Organizational Information Requirements, Media Richness and Structural Design. *Management Science* 32, 5: 554–571.
14. R. Fruchter and Samuel Yen. RECALL in Action. *Computing in Civil and Building Engineering (2000)*, ASCE, 1012–1020.

15. Barney G Glaser and Anselm L Strauss. 1967. The discovery of grounded theory. *International Journal of Qualitative Methods* 5: 1–10.
16. Gene Golovchinsky and Laurent Denoue. 2002. Moving markup: Repositioning Freeform Annotations. *Proceedings of the 15th annual ACM symposium on User interface software and technology - UIST '02*, ACM Press, 21.
17. Jonathan Grudin. 1988. Why CSCW applications fail: problems in the design and evaluationof organizational interfaces. *Proceedings of the 1988 ACM conference on Computer-supported cooperative work*, ACM Press, 85–93.
18. Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. *Advances in Psychology* 52, C: 139–183.
19. Khe Foon Hew and Wing Sum Cheung. 2013. Audio-based versus text-based asynchronous online discussion: Two case studies. *Instructional Science* 41, 2: 365–380.
20. Debby Hindus and Chris Schmandt. 1992. Ubiquitous audio: Capturing Spontaneous Collaboration. *Proceedings of the 1992 ACM conference on Computer-supported cooperative work - CSCW '92*, ACM Press, 210–217.
21. Philip S. Holzman and Clyde Rousey. 1966. The voice as a percept. *Journal of Personality and Social Psychology* 4, 1: 79–86.
22. Inc. IBM. IBM Bluemix.
23. G. J. Jin, Z., Finkelstein, A., DiVerdi, S., Lu, J., and Mysore. 2016. CUTE: a concatenative method for voice conversion using exemplar-based unit selection. *In 41st IEEE International Conference on Acoustics Speech and Signal Processing*, IEEE.
24. Qin Jin, Arthur R. Toth, Tanja Schultz, and Alan W. Black. 2009. Speaker de-identification via voice transformation. *Proceedings of the 2009 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2009*, 529–533.
25. Juho Kim, Elena L. Glassman, Andrés Monroy-Hernández, and Meredith Ringel Morris. 2015. RIMES: Embedding Interactive Multimedia Exercises in Lecture Videos. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*, ACM, 1535–1544.
26. Stephen R Levine and Susan F Ehrlich. 1991. The Freestyle System. In *Human-Machine Interactive Systems*. Springer, 3–21.
27. Guang Li, Xiang Cao, Sergio Paolantonio, and Feng Tian. 2012. SketchComm. *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work - CSCW '12*, ACM Press, 359.
28. J Lofland and L Lofland. 1971. Analyzing social settings. *Belmont, CA: Wadsworth*.
29. Philip Marriott. 2002. Voice vs text-based discussion forums: An implementation of Wimba Voice Boards. *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, 640–646.
30. Joseph A Maxwell. 2013. Qualitative research design: An interactive approach. In *Qualitative research design: An interactive approach*. 23–38.
31. Cosmin Munteanu, Ronald Baecker, Gerald Penn, Elaine Toms, and David James. 2006. The effect of speech recognition accuracy rates on the usefulness and usability of webcast archives. *Proc. CHI '06, ACM Press*: 493-502.
32. Eugene W. Myers. 1986. An O(ND) difference algorithm and its variations. *Algorithmica* 1, 1–4: 251–266.
33. Yingxin Pan, Danning Jiang, Michael Picheny, and Yong Qin. 2009. Effects of real-time transcription on non-native speaker's comprehension in computer-mediated communications. *Proceedings of the 27th international conference on Human factors in computing systems - CHI '09*, 2353-2356.
34. Jennifer Pearson, Simon Robinson, and Matt Jones. 2015. PaperChains: Dynamic Sketch+Voice Annotations. *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15*, ACM, 383–392.
35. Jeff Raskin. 2000. *The Humane Interface: New Directions for Designing Interactive Systems*.
36. Steve Rubin, Floraine Berthouzoz, Gautham J. Mysore, Wilmot Li, and Maneesh Agrawala. 2013. Content-based tools for editing audio stories. *Proceedings of the 26th annual ACM symposium on User interface software and technology - UIST '13*, ACM Press, 113–122.
37. Michael F. Scheier and Charles S. Carver. 1985. The Self-Consciousness Scale: A Revised Version for Use with General Populations. *Journal of Applied Social Psychology 15*, 687–699.
38. C Schmandt. 1981. The intelligent ear: A graphical interface to digital audio. *... , IEEE International Conference on Cybernetics and ...*.
39. Jeremiah Scholl, John McCarthy, and Rikard Harr. 2006. A comparison of chat and audio in media rich environments. *Proceedings of ACM CSCW'06 Conference on Computer-Supported Cooperative Work*: 323–332.
40. Marc Schröder. 2001. Emotional Speech Synthesis: A Review. *INTERSPEECH*, 561–564.

41. Venkatesh Sivaraman, Dongwook Yoon, and Piotr Mitros. 2016. Simplified Audio Production in Asynchronous Voice-Based Discussions. *Proceedings of the 2016 ACM Conference on Human Factors in Computing Systems*.
42. Hagen Soltau, George Saon, and Tara N. Sainath. 2014. Joint training of convolutional and non-convolutional neural networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 5572–5576.
43. LA Stark, S Whittaker, and J Hirschberg. 2000. ASR satisficing: the effects of ASR accuracy on speech retrieval. *INTERSPEECH*.
44. Lisa Stifelman, Barry Arons, and Chris Schmandt. 2001. The Audio Notebook: Paper and Pen Interaction with Structured Speech. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM Press, 182–189.
45. Michael Tsang, George W Fitzmaurice, Gordon Kurtenbach, Azam Khan, and Bill Buxton. 2002. Boom Chameleon: Simultaneous Capture of 3D Viewpoint, Voice and Gesture Annotations on a Spatially-aware Display. *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, ACM, 111–120.
46. H. Valbret, E. Moulines, and J. P. Tubach. 1992. Voice transformation using PSOLA technique. *Speech Communication* 11, 2–3: 175–187.
47. Sunil Vemuri, Philip DeCamp, Walter Bender, and Chris Schmandt. 2004. Improving speech playback using time-compression and speech recognition. *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04*, ACM Press, 295–302.
48. QianYing Wang and Clifford Nass. 2005. Less Visible and Wireless: Two Experiments on the Effects of Microphone Type on Users' Performance and Perception. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 809–818.
49. Steve Whittaker and Brian Amento. 2004. Semantic speech editing. *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04*, ACM Press, 527–534.
50. Steve Whittaker, Erik Geelhoed, and Elizabeth Robinson. 1993. Shared workspaces: how do they work and when are they useful? *International Journal of Man-Machine Studies* 39, 5: 813–842.
51. Steve Whittaker, Julia Hirschberg, Brian Amento, et al. 2002. SCANMail: a voicemail interface that makes speech browsable, readable and searchable. *Proceedings of the SIGCHI conference on Human factors in computing systems Changing our world, changing ourselves - CHI '02*, ACM Press, 275.
52. Steve Whittaker, Patrick Hyland, and Myrtle Wiley. 1994. Filochat: handwritten notes provide access to recorded conversations. *Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94*, ACM Press, 271–277.
53. Lynn D. Wilcox, Bill N. Schilit, and Nitin Sawhney. 1997. Dynomite: a dynamically organized ink and audio notebook. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*, ACM Press, 186–193.
54. Dongwook Yoon, Nicholas Chen, François Guimbretière, and Abigail Sellen. 2014. RichReview: Blending Ink, Speech, and Gesture to Support Collaborative Document Review. *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ACM, 481–490.
55. Dongwook Yoon, Nicholas Chen, Bernie Randles, et al. 2016. RichReview++: Deployment of a Collaborative Multi-modal Annotation System for Instructor Feedback and Peer Discussion. *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, 195–205.