SPIRE-ABC: An online tool for acoustic-unit boundary correction (ABC) via crowdsourcing

Chiranjeevi Yarra
Electrical Engineering
Indian Institute of Science
Bangalore 560012, India
chiranjeeviy@iisc.ac.in

Kausthubha N K
Electrical Engineering
Indian Institute of Science
Bangalore 560012, India
kausthubha12@gmail.com

Prasanta Kumar Ghosh
Electrical Engineering
Indian Institute of Science
Bangalore 560012, India
prasantg@iisc.ac.in

***Abstract*—Time aligned acoustic-unit (AU) boundaries are often important for the applications related to human computer interaction (HCI). These boundaries are typically estimated using automatic speech recognizer (ASR). However, they are often erroneous due to mistakes by the ASR. In general, the boundaries estimated using ASR are corrected manually, however, the manual correction is cumbersome, time consuming and costly. Crowdsourcing has been known to be effective in such scenarios via online tool which allows an interactive interface that can be used by a large number of annotators. Typically, the annotators are non-expertise in speech specific knowledge such as spectrograms. Thus, it is required a customized interface to yield better performance from non-expert annotators. We propose an online tool called SPIRE-ABC for correcting noisy AU boundaries, for example, estimated using ASR. SPIRE-ABC could be useful to operate in a crowdsourcing environment with or without crowdsourcing platforms. This is developed using JavaScript and a customizable audio waveform visualization interface called WaveSurfer. Currently SPIRE-ABC is available in two versions – 1) an online interface, in which users can do the correction by uploading an audio file and the corresponding noisy AU boundaries; 2) an offline (downloadable) version for setting up into users' web-server. We perform experiments where annotators are asked to use SPIRE-ABC to correct the AU boundaries (syllable and word) obtained from Kaldi ASR on TIMIT speech data. We find improvements in the AU boundaries following manual correction using the SPIRE-ABC and the improvements obtained based on non-expert annotators are similar to those based on expert annotators.**

***Index Terms*—Acoustic unit boundary correction, Crowdsourcing, Manual correction, Online interface**

I. Introduction

Time aligned boundaries of AUs – phonemes, syllables and words – are useful in the applications related to human computer interaction (HCI) and computer assisted language learning (CALL) such as automatic foreign language pronunciation tutoring [1] and automatic detection of mispronunciation [2]. Typically these boundaries are estimated from automatic speech recognizer (ASR) either using forced alignment or by doing recognition [2]. However, these boundaries often suffer from errors due to inaccuracies in the ASR systems [2]. These errors in the boundaries are typically corrected with human intervention. Apart from the pronunciation evaluation system [3], the manually corrected boundaries are useful in concatenative speech synthesis, phonetics [4] and in developing a quality speech corpus with time aligned AU boundaries [5]. However, manual correction is time-consuming and expensive [6], [7].

Crowdsourcing has been found to be a cost effective solution in speech language applications [8]. Gao et al. have shown that the crowdsourcing workers together outperformed the experts on word alignment tasks [9]. Typically in crowdsourcing, tasks are distributed to a group of people for completion, such that the costs involved in the task completion can be reduced [8]. For achieving a lower cost, crowdsourcing methods provide a way for non-experts to complete the tasks that would normally be reserved for experts [10]. In general, most of the crowdsourcing tasks are performed via online tools using specific platforms that allow communication between requesters and workers [10], for example Amazon Mechanical Turk (AMT) [11]. Correction of acoustic-unit (AU) boundaries, in particular, requires development of a customized interface where the workers can listen to the audio over any period of time by selecting parts of the audio waveform as well as edit the AUs and/or their boundaries. In this correction process, the cost effective solutions can be achieved by considering the workers, who are not necessarily having expertise in speech domain. Thus, while developing the interface, it is not necessarily to incorporate any speech specific visualization tool such as spectrograms. In addition, incorporating such tools could cause confusion to the non-expert workers. Considering these, in this work, we develop an online tool called SPIRE-ABC to correct the noisy AU boundaries (e.g., obtained from ASR) via crowdsourcing and that can be operated with or without web platforms similar to AMT.

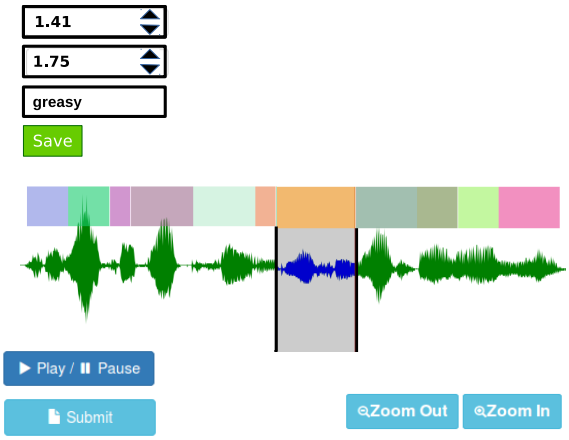

Fig. 1. Annotation Interface of the SPIRE-ABC

SPIRE-ABC reads an audio (.wav format) as well as corresponding noisy AU boundaries (.csv format); and creates an audio-visual interface for AU boundary correction (ABC) by the users. Figure 1 shows SPIRE-ABC annotation interface for an exemplary speech segment (*she had your dark suit in greasy*

wash water all the year) taken from the TIMIT corpus [12]. The interface displays the audio waveform as well as noisy AU (word) boundaries of the speech segment. The noisy word boundaries corresponding to the speech segment are shown in Table I. In the table, there are eleven words along with their start and end times. All word segments are displayed with colored rectangular boxes on top of the waveform display. Each rectangular box is placed according to the start time of a word with the width equals to the word duration along the time axis. The word boundaries can be modified with the mouse click. On a mouse click, a box with adjustable boundaries is displayed on the waveform (gray shaded region with black thick boundaries in Figure 1). At the same time, the words' start, end times and transcription are displayed on top left corner (black solid rectangular boxes in Figure 1). The word boundaries are modified by changing the boundaries of the gray box and saved by clicking the save button. For providing better clarity in the annotation two zoom buttons ('Zoom out' and 'Zoom in') are provided. After complete annotation of all word boundaries, modified word boundaries are updated into local storage with a click on the 'submit' button.

TABLE I
AN EXEMPLARY SET OF NOISY AU BOUNDARIES CORRESPONDING TO THE SPEECH SEGMENT *she had your dark suit in greasy wash water all the year*.

Sl no	Start time	End time	Word	Sl no	Start time	End time	Word	Sl no	Start time	End time	Word
1	0.19	0.35	she	5	1.02	1.32	suit	9	2.08	2.34	water
2	0.35	0.64	had	6	1.32	1.41	in	10	2.34	2.51	all
3	0.64	0.71	your	7	1.41	1.75	greasy	11	2.51	2.78	year
4	0.71	1.02	dark	8	1.75	2.08	wash				

The functional modules in the SPIRE-ABC are written in JavaScript, developed based on WaveSurfer [13], which is a customized audio waveform visualization JavaScript. The WaveSurfer has been used in many audio-video interface applications. Saiz et al. have used the same in a web platform for introducing user interface (UI) controls to compose and combine the audio streams [14]. Baker et al. have used it to create an analysis platform for recordings of wild life sounds via crowdsourcing for removing voice introductions, identifying external noises and separating two species in the recordings [15]. Hsu has used WaveSurfer for syncing voice-over narrated audios and handwritten lecture videos to record and edit the online educational lectures [16]. Wang has used it to record and edit the audios using web platform [17]. Matuszewki et al. have used to create a set of online components for interactive audiovisual rendering of audio signals [18]. Buffa et al. have used WaveSurfer to create online multi-track audio player for musicians [19]. However, WaveSurfer is a general purpose JavaScript; hence, it is required to customize it according to the SPIRE-ABC functionalities.

WaveSurfer JavaScript is also modified to incorporate additional functionalities required by the SPIRE-ABC. An automatic script is written for creating JavaScript object notation (JSON) file from the csv file containing noisy AU boundaries. SPIRE-ABC, currently, can be used either in a online mode

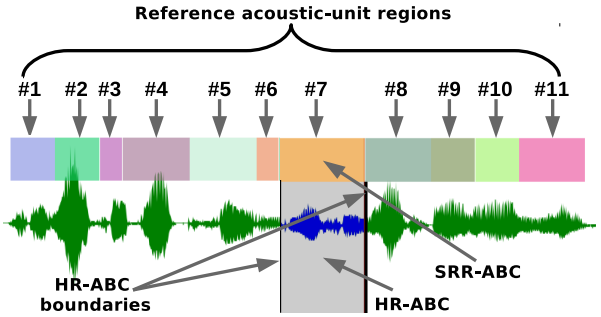

Fig. 2. An exemplary figure describing the proposed annotation interface in SPIRE-ABC

from our web sever or by downloading source script to set-up in users' web server. The usage of SPIRE-ABC for both cases is provided in this report as well as available in the SPIRE-ABC web page with the detailed instructions. Experiments are conducted to know the effectiveness of SPIRE-ABC considering syllable and word boundaries obtained from Kaldi ASR on TIMIT speech data in online mode. Following the manual correction using SPIRE-ABC, the obtained AU boundaries results in better values interms of three objective measures, namely, mean absolute difference (MAD), correct alignment rate (CAR) and overlap rate (OVR) compared to those values obtained with the estimated boundaries from ASR. Further, the improvements in all three measures are similar when the manual correction is obtained either from non-expert or expert annotators.

II. SPIRE-ABC

Figure 2 shows a part of the SPIRE-ABC annotation interface for the exemplary speech segment used in Figure 1. It displays two different types of region markings on the waveform for ABC. The first type is formed by the segment between every two consecutive boundaries and is referred to as reference AU region, shown by eleven colored rectangular regions in Figure 2 with corresponding serial number of the words from Table I. The second type is created by selecting a reference region with a mouse click and is referred as a highlighted AU region, shown by gray shaded portion corresponding to the 7-th word unit in the figure. Note that in the SPIRE-ABC, the selected reference regions can not be edited by changing the colored rectangular boxes directly for ABC; however, it can be done via highlighted regions. Hence, SPIRE-ABC provides separate control options to the highlighted region for ABC (HR-ABC). On mouse click, the audio of the HR-ABC is played by the interface. SPIRE-ABC provides a separate zoom control to the HR-ABC for better audio waveform visualization; and it also facilitates resizing HR-ABC by dragging the boundaries in both directions for correction of boundaries in an annotator-friendly manner. When the modified HR-ABC is saved using a 'save' option as shown in Figure 1, it automatically updates the corresponding selected reference region for ABC (SRR-ABC).

The SPIRE-ABC functionalities have been developed by introducing modifications as well as additions in the WaveSurfer JavaScript. WaveSurfer JavaScript is also available with addi-

tional plug-in called region plug-in (RPI) to create reference regions using a JSON file containing reference regions details. The RPI also provides extra functionalities such as dragging and resizing of the reference regions, which, however, are required only for HR-ABC and not for SRR-ABC. So in the case of SRR-ABC, those functions are disabled. On the other hand, RPI, by default, creates HR-ABC for all reference regions while it is required to create HR-ABC for only one region corresponding to the SRR-ABC that an annotator intends to edit. This is resolved by appropriately modifying both the WaveSurfer and the RPI JavaScripts. Another major challenge in WaveSurfer with RPI is the incorporation of the save controls for updating SRR-ABC using HR-ABC. This is because RPI always holds one UI handler corresponding to a mouse click; i.e., the UI handler points towards either SRR-ABC or HR-ABC but not both at a time. Therefore, the UI handler pointer to SRR-ABC is lost when HR-ABC is modified. Hence using existing WaveSurfer plus RPI set-up, it is not possible to update SRR-ABC by modifying HR-ABC. This is resolved by adding new control variables in both the WaveSurfer and the RPI JavaScripts.

III. Components of SPIRE-ABC

The components involved in the proposed SPIRE-ABC annotation interface are described with the help of a block diagram in Figure 3. The block diagram has two major stages: 1) data preparation, 2) data annotation. The data preparation stage automatically creates JSON file from the csv file containing noisy AU boundaries (reference regions). The data annotation stage creates interface for manual correction and it has five components. The first component displays the waveform of the audio and reference regions. The second component creates HR-ABC from SRR-ABC on a mouse click. The third component allows resizing HR-ABC to modify its boundaries. The fourth component controls zooming and playing of the audio segment in HR-ABC. The last component updates SRR-ABC with the modified HR-ABC.

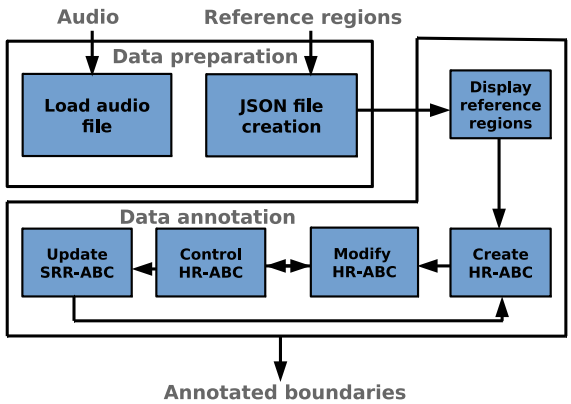

Fig. 3. Block diagram describing the main components in SPIRE-ABC

Figure 4 shows the detailed modifications in the proposed SPIRE-ABC with respect to the WaveSurfer. In the figure, solid black rectangular boxes indicate functional blocks involved in each component of SPIRE-ABC. Black dotted square box indicates the WaveSurfer and the boxes (blue) within that dotted box are the existing functional blocks in the WaveSurfer – Play, Zoom, Regions plug-in (RPI) and Save regions – that are customized to the need of SPIRE-ABC. The proposed modifications to the WaveSurfer are indicated by the blue boxes outside the dotted square box. The detailed description of each functional block is discussed in the following sub-sections.

A. JSON file creation

JSON file is a light weight text-based open standard designed for human-readable data interchange. It is in text format that is typically programming language independent but uses conventions that are familiar to programmers of the C-language family, including C, C++, C#, Java, JavaScript, Perl, Python etc. JSON syntax is easy to use and has wide range of browser compatibility. WaveSurfer requires the reference regions as well as the AU transcriptions in JSON format as collection of objects. Each AU boundary is represented as an object, containing values of start time, end time and the transcription. Table II shows an exemplary conversion of AU boundaries from csv to JSON format. From the table, it is observed that each object is represented using the braces and each value in the object is separated by commas.

TABLE II
AN EXEMPLARY CONVERSION OF CSV TO JSON FORMAT FOR TWO WORD BOUNDARIES.

csv format	JSON format
start, end, word 0.19, 0.35, she 0.35, 0.64, had	[{“start”:“0.19”,“end”:“0.35”,“data” :{“word”:“she”}},{“start”:“0.35”, “end”:“0.64”,“data”:{“word”:“had”}}]

B. WaveSurfer

WaveSurfer is a customized audio waveform visualization built on top of Web Audio API and HTML5 canvas. The Web Audio API provides a powerful and versatile system for controlling audio on the Web, allowing to choose audio sources, adding effects to audio and creating audio visualizations [20]. The HTML5 canvas is used to draw graphics via JavaScript for real-time visualization [21]. WaveSurfer displays and plays the audio segments associated with reference regions using RPI. The WaveSurfer and RPI update the regions according to the annotator’s modification and register the callbacks associated with the various UI actions such as clicking, resizing and moving of the regions.

C. Display

SPIRE-ABC displays waveform along with reference regions. These regions can’t be displayed using WaveSurfer alone, it requires an additional RPI plug-in. In addition to creating the regions, RPI plug-in can also displays three additional control functions to WaveSurfer – 1) play the audio of each region 2) resize each region by dragging its boundaries 3) move the entire region. In SPIRE-ABC, we obtain AU boundaries with WaveSurfer and RPI using only the first control function and disabling remaining control functions. This is because the reference region boundaries are not allowed for direct changes from the annotator; updating

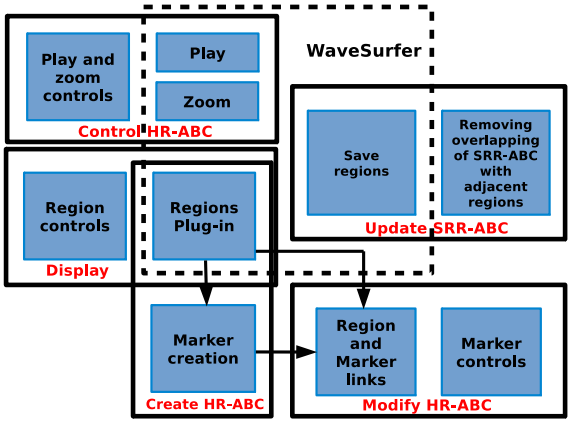

Fig. 4. Block diagram summarizing the modifications made on the wavesurfer functionalities required for SPIRE-ABC

those is allowed through the changes made in HR-ABC. In the Figure 2, it is observed that the reference regions have lower height than the HR-ABC and are displayed above the waveform.

D. Select boundary

SPIRE-ABC creates HR-ABC following annotator's mouse click on an SRR-ABC. This is done in two steps by enabling the marker functionalities that are available in the RPI. In the first step, all the reference regions are highlighted using marker enabled RPI with the three additional region control functions – play, resize and move. The second step disables all the reference regions except the SRR-ABC. Hence HR-ABC is created. This is performed by passing region-id of the selected reference region to the highlighted reference regions.

E. Adjust boundary

Region-id contains the information of only one region (i.e.,start time, end time, transcription of region) which is activated by the mouse click from a user. With the region-id, the RPI allows modification and updates each region after the modification. This, in turn, requires each region to be separately controlled with the existing WaveSurfer and RPI. However in the SPIRE-ABC, modifications of the SRR-ABC are performed through the changes made in HR-ABC. Hence, it is required to store the callbacks of SRR-ABC while modifying the HR-ABC. But in the existing WaveSurfer and RPI setup, the region-id can point to either SRR-ABC or HR-ABC but not both. To overcome this, WaveSurfer and RPI set-up is modified by defining two functions – main function and a sub-function. The main function is activated with the mouse click on the SRR-ABC and updates SRR-ABC through a globally declared variables. The sub-function is a part of the main function and it is activated with the mouse click on the HR-ABC and the changes made in the sub-function are updated in the global variables. Hence, the modifications done in HR-ABC automatically update the SRR-ABC.

F. Control boundary

The annotation process may require the user to view the waveform (or parts of the audio) at different zoom levels in order to accurately mark the AU boundaries particularly for finer speech units such as phonemes and syllables. It is observed that the annotation process is more convenient with finite number of zoom levels compared to continuous zooming with slide bar control. However in WaveSurfer, zooming functionality is available only with the sliding bar control. In the SPIRE-ABC, we replace the slide bar based zooming as in WaveSurfer with two buttons ('Zoom In' and 'Zoom Out' in Figure 1) for zooming with four discrete zoom levels ($\frac{1}{2}$x, $\frac{1}{4}$x, $\frac{1}{8}$x and $\frac{1}{16}$x).

G. Save boundary

The start time of an AU is same as the end time of its left neighboring AU. Similarly the end time of the same AU is equal to its right neighboring AU. This indicates that the start and end boundaries of any reference region depend on its neighboring reference regions. Hence, the changes made in the boundaries of any SRR-ABC would reflect in its neighboring reference regions automatically. However, implementation with RPI saves only the SRR-ABC boundaries specified by the region-id and this boundary information could not be passed to its neighboring reference regions. To circumvent this problem, we store all the reference regions information in a variable and update the variable entries corresponding to the SRR-ABC and its neighboring reference regions according to the changes made in the HR-ABC.

IV. Usage of SPIRE-ABC

SPIRE-ABC is available in two versions[1] – online and offline (downloadable). In online mode users are requested to upload both audio file (currently supporting .wav format) to be annotated and noisy AU boundaries (in .csv format) in a format similar to the sample shown Table II. The sampled files are also available in SPIRE-ABC web page. Once both files are successfully uploaded in the required format, SPIRE-ABC annotation interface is activated. Using this interface, users can modify the uploaded noisy AU boundaries by following the instructions that are provided on the SPIRE-ABC web page, which are also presented in the following sub-section. Finally, the modified boundaries can be downloaded in the a format (.csv) identical to the one in which the original noisy boundaries are uploaded.

In offline (downloadable) mode, users can download SPIRE-ABC JavaScript from the web page to perform the annotation through their own web server[2]. In this case, users need to place the JavaScript in their web server and update the source paths of the audio files and the csv files[3] containing noisy AU boundaries. Users also need to update the destination paths for the output csv files containing updated ABCs as modified by the annotator. The SPIRE-ABC automatically retrieves each file pair (audio and csv) from the source location, activates

[1] http://spire.ee.iisc.ernet.in/spire-abc/

[2] we have so far successfully tested using LAMP (Linux, Apache, MySQL, PHP) stack on UBUNTU 14.04 operating system.

[3] names of both the audio and the csv files (excluding extensions) should be identical

the annotation interface for users to modify the boundaries and places the annotated ABCs in the destination location.

A. Instructions for ABC

The instructions involved in ABC are provided below. These instructions are required for using the SPIRE-ABC annotation interface (Figure 1). These instructions are identical for both online and offline modes of SPIRE-ABC.

1) Click on play/pause button to listen the entire audio segment.
2) Click on a reference region to be modified (SRR-ABC). On click, SPIRE-ABC displays the corresponding HR-ABC. It also displays the time aligned boundaries as well as transcription of HR-ABC on the top left corner, as indicated by black rectangular boxes in Figure 1.
3) Click on either SRR-ABC or HR-ABC to listen the audio segments corresponding to the respective reference regions.
4) Click on play controls to change the play speed of HR-ABC.
5) Click on zoom controls to change the display of HR-ABC.
6) Resize or drag the HR-ABC boundaries to modify SRR-ABC.
7) Click the save button to update SRR-ABC with modified HR-ABC.
8) Repeat 2nd to 7th steps for any other SRR-ABC.
9) Click submit to save the modified AU boundaries into the output file.

V. Experiments and results

A. Experimental set-up

We consider three measures for computing the accuracy of the AU boundaries corrected using SPIRE-ABC. The first measure is mean absolute difference (MAD) between the ground truth and the corrected AU boundaries [22]. The second measure is the percentage of AU boundaries that fall within a tolerance of 40ms from the ground truth AU boundaries [23], referred to as correct alignment rate (CAR). The third measure is the overlap rate (OVR) defined in [24]. OVR measures the amount of overlap between the corrected and ground truth segments for all AUs in the sentences considered for ABC. Thus, for a set of accurate AU boundaries, MAD should be low while CAR and OVR should be high. We perform the experiments in an online mode. The annotators are asked to follow the instructions provided in the online portal and during correction, the annotators are not provided with any further instructions. We select nine annotators belonging to three categories each containing three annotators. Annotators in the first category have prior experience in doing correction at various AU levels, called expert annotator (EA). Annotators in the second category do not have any experience in the correction task; however, they have experience in handling the annotated speech data, called inexperienced annotator (IEA). Annotators in the third category have experience neither in speech data handling nor in correction, called new annotator (NA). Each annotator is asked to correct the AUs at two levels – syllable and word. All the chosen annotators are either post graduate or doctoral students at the institute. For the correction task, each annotator is given a fixed amount of cash as a token of appreciation.

B. Results and discussions

For the correction, we use 30 phonetically balanced audio files from the test set of the TIMIT [12] corpus. We consider phoneme and word aligned boundaries corresponding to each audio file from the corpus as the ground truth. We convert the phoneme boundaries into syllable boundaries using "NIST tsylab" syllabification software [25]. Noisy AU boundaries are obtained from Kaldi speech recognition tool kit [26] under two forced alignment setups using deep neural network based acoustic models (Karels' implementation). The first setup uses the acoustic models learnt from the fisher English (FE) [27] data for the forced alignment of the TIMIT data. This setup is useful to analyze the variabilities in ABC when noisy AU boundaries are obtained from the forced alignment with a mismatch between the train and test sets. The second setup considers a matched scenario in which acoustic models are learnt from the TIMIT train data. Considering both variants of each AU, nine annotators correct a total of 120 files – 30 files containing word AUs aligned with FE acoustic model (denoted by FE_W), 30 files containing syllable AUs aligned with FE acoustic model (denoted by FE_S), 30 TIMIT_W files, and 30 TIMIT_S files. Entire set of 120 files is used to create nine subsets, each containing 52 files for one annotator. Each of FE_W, FE_S, TIMIT_W, and TIMIT_S groups is randomly divided in three subsets each containing 10 files. Three non-overlapping sets (S_1, S_2, S_3) of 40 files are obtained by combining one subset from each group. In addition to this, four files from each of S_1, S_2, and S_3 are randomly chosen to construct a common set (S_c) containing 12 files. S_c is combined with each of S_1, S_2, and S_3 to have three sets of 52 files, which are given to three EAs as well as three IEAs and NAs separately.

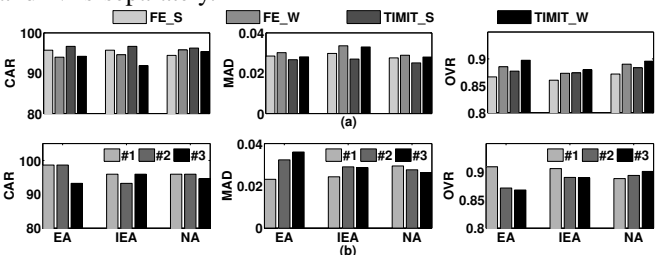

Fig. 5. CAR, MAD and OVR measures obtained after manual correction for all three annotator categories under four conditions – FE_S, FE_W, TIMIT_S and TIMIT_W.

Figure 5a shows the CAR, MAD and OVR obtained under FE and TIMIT setups for syllable and word boundaries after manual correction by the EAs, IEAs and NAs. Before the manual correction using SPIRE-ABC (i.e., boundaries from forced alignment directly), CAR is found to be 83.02, 78.92, 86.72 & 82.35 for FE_S, FE_W, TIMIT_S & TIMIT_W respectively. Similarly, MAD and OVR are found to be 0.0465, 0.0518, 0.0352 & 0.0394 and 0.7927, 0.8120, 0.8257 & 0.8398

respectively. From the figure, it is observed that the CAR and OVR are higher and MAD is lower after the manual correction using SPIRE-ABC compared to those from forced alignment directly. This indicates the benefit of using SPIRE-ABC for manual correction of AU boundaries by annotators from all three categories. It is also observed that the performance measures obtained by NAs are not significantly different from those by EAs and IEAs. This suggests that all three type of annotators could produce ABC with similar quality using the SPIRE-ABC interface. From the figure, the OVRs are found to be 0.8663, 0.8830, 0.8783 and 0.8910 for FE_S, FE_W, TIMIT_S, and TIMIT_W respectively when averaged across all three annotator categories. These OVR values are more than misalignment OVR threshold (0.75) as suggested by Paulo et al. [24]. These observations indicate that the SPIRE-ABC interface is effective for ABC using crowdsourcing.

From Figure 5a, it is also observed that the CAR, MAD and OVR obtained from NAs are higher (but not significantly) compared to EAs and IEAs for some specific conditions. For example, NAs have higher CAR than both EAs and IEAs for TIMIT_S setup. We investigate this using the performance of each annotator on the common set (S_c). Figure 5b shows the CAR, MAD and OVR obtained from each annotator of all three categories. From the figure, it is observed that the first annotator in EA category (EA#1) shows better performance across all performance measures over IEAs and NAs. However, interestingly, the EA#3 has lower performance among all EAs and across both the IEAs and NAs. This suggests that the SPIRE-ABC is annotator-friendly irrespective of the annotator category and the performance variabilities, as observed in Figure 5a, mainly depend on the AU correction ability of the annotator.

VI. Conclusions

In this work, we present SPIRE-ABC that helps in correcting errors in noisy acoustic-unit boundaries using web interface via crowdsourcing. This is developed by creating additional functional modules as well as modifying the existing functional modules in the WaveSurfer JavaScript. SPIRE-ABC is available in online as well as in downloadble offline versions along with the instructions for using ABC. Experiments on TIMIT corpus have shown improvements in the AU boundaries after manual correction of those obtained from forced alignment using SPIRE-ABC online version. Further works are required for adding all reference acoustic-unit transcriptions as well as displaying a 2-D representation of audio signal like spectrogram along with the waveform for better clarity during annotation.

References

[1] M. Eskenazi, "Using automatic speech processing for foreign language pronunciation tutoring: Some issues and a prototype," *Language learning & technology*, vol. 2, no. 2, pp. 62–76, 1999.

[2] H. Franco, L. Neumeyer, M. Ramos, and H. Bratt, "Automatic detection of phone-level mispronunciation for language learning." *Proc of EUROSPEECH*, 1999.

[3] S. Witt and S. Young, "Performance measures for phone-level pronunciation teaching in call," *Proc. of the Workshop on Speech Technology in Language Learning*, pp. 99–102, 1998.

[4] K. Sjölander, "An HMM-based system for automatic segmentation and alignment of speech," *Proc of Fonetik*, vol. 2003, pp. 93–96, 2003.

[5] M. A. Pitt, K. Johnson, E. Hume, S. Kiesling, and W. Raymond, "The Buckeye corpus of conversational speech: Labeling conventions and a test of transcriber reliability," *Speech Communication*, vol. 45, no. 1, pp. 89–95, 2005.

[6] T. J. Hazen, "Automatic alignment and error correction of human generated transcripts for long speech recordings." *Proc of INTERSPEECH*, pp. 1606–1609, 2006.

[7] J. Yuan, N. Ryant, M. Liberman, A. Stolcke, V. Mitra, and W. Wang, "Automatic phonetic segmentation using boundary models." *Proc of INTERSPEECH*, pp. 2306–2310, 2013.

[8] M.-C. Yuen, I. King, and K.-S. Leung, "A survey of crowdsourcing systems," *Privacy, Security, Risk and Trust (PASSAT) and IEEE Third Inernational Conference on Social Computing (SocialCom)*, pp. 766–773, 2011.

[9] Q. Gao and S. Vogel, "Consensus versus expertise: A case study of word alignment with mechanical turk," *Proceedings of the NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pp. 30–34, 2010.

[10] G. Parent and M. Eskenazi, "Speaking to the crowd: Looking at past achievements in using crowdsourcing for speech and predicting future challenges." *Proc of INTERSPEECH*, pp. 3037–3040, 2011.

[11] M. Marge, S. Banerjee, and A. I. Rudnicky, "Using the amazon mechanical turk for transcription of spoken language," *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 5270–5273, 2010.

[12] V. Zue, S. Seneff, and J. Glass, "Speech database development at MIT: TIMIT and beyond," *Speech Communication*, vol. 9, no. 4, pp. 351–356, 1990.

[13] Katspaugh, "wavesurfer.js," *Available online: http://wavesurfer-js.org/*, 2012.

[14] V. Saiz, B. Matuszewski, and S. Goldszmidt, "Audio oriented ui components for the web platform," *Proceedings of WAC 1st Web Audio Conference January 26-28, Paris, France*, 2015.

[15] E. Baker, B. W. Price, S. D. Rycroft, J. Hill, and V. S. Smith, "BioAcoustica: a free and open repository and analysis platform for bioacoustics," *Database*, pp. 1–10, 2015.

[16] A. Hsu, "Creating and editing" Digital Blackboard" videos using Pentimento: with a focus on syncing audio and visual components," Ph.D. dissertation, Massachusetts Institute of Technology, 2015.

[17] J. Wang, "Pentimento: non-sequential authoring of handwritten lectures," Ph.D. dissertation, Massachusetts Institute of Technology, 2015.

[18] B. Matuszewski, N. Schnell, and S. Goldszmidt, "Interactive audiovisual rendering of recorded audio and related data with the WavesJS building blocks," *Proceedings of the 2nd Web Audio Conference (WAC-2016), Atlanta*, 2016.

[19] M. Buffa, A. Hallili, and P. Renevier, "MT5: a HTML5 multitrack player for musicians," *Proceedings of WAC 1st Web Audio Conference January 26-28, 2015-IRCAM & Mozilla Paris, France*, 2015.

[20] B. Smus, *Web audio API*. O'Reilly Media, Inc., 2013.

[21] S. Fulton and J. Fulton, *HTML5 canvas*. O'Reilly Media, Inc., 2013.

[22] J. Adell, A. Bonafonte, J. A. Gómez, and M. J. Castro, "Comparative study of automatic phone segmentation methods for TTS." *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 309–312, 2005.

[23] S. Brognaux and T. Drugman, "HMM-based speech segmentation: improvements of fully automatic approaches," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 1, pp. 5–15, 2016.

[24] S. Paulo and L. C. Oliveira, "Automatic phonetic alignment and its confidence measures," *Advances in Natural Language Processing*, pp. 36–44, 2004.

[25] B. Fisher, "tsylb2-1.1: syllabification software," *National Institute of Standards and Technology, Available online: https://www.nist.gov/itl/iad/mig/tools, last accessed on 07–09–16*, 1996.

[26] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," *IEEE workshop on automatic speech recognition and understanding (ASRU)*, 2011.

[27] C. Cieri, D. Miller, and K. Walker, "The Fisher Corpus: a resource for the next generations of speech-to-text." *4th international conference on Language Resources Evaluation*, vol. 4, pp. 69–71, 2004.