INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Using Clinician Annotations to Improve Automatic Speech Recognition of Stuttered Speech

Peter A. Heeman,[1,2] *Rebecca Lunsford,*[1,2] *Andy McMillin,*[2,3] *and J. Scott Yaruss*[4]

[1]BioSpeech, Lake Oswego, OR
[2]Center for Spoken Language Understanding, Oregon Health & Science University, Portland, OR
[3]Speech & Hearing Sciences, Portland State University, Portland, OR
[4]Dept. of Communication Sciences and Disorders, University of Pittsburgh, Pittsburgh, PA

heemanp@ohsu.edu

Abstract

In treating people who stutter, clinicians often have their clients read a story in order to determine their stuttering frequency. As the client is speaking, the clinician annotates each disfluency. For further analysis of the client's speech, it is useful to have a word transcription of what was said. However, as these are real-time annotations, they are not always correct, and they usually lag where the actual disfluency occurred. We have built a tool that rescores a word lattice taking into account the clinician's annotations. In the paper, we describe how we incorporate the clinician's annotations, and the improvement over a baseline version. This approach of leveraging clinician annotations can be used for other clinical tasks where a word transcription is useful for further or richer analysis.

Index Terms: stuttering, automatic speech recognition, disfluency counts, user-interface

1. Introduction

Stuttering is a communication disorder characterized by disfluencies that are frequent and disruptive to communication. Common types of disfluencies include sound, word, and phrase repetitions, revisions, prolongations, blocks, and interjections. Speech Language Pathologists (SLPs or clinicians) use disfluency counts to decide whether a client should be treated, to assess treatment progress, and to document treatment outcomes [1, 2, 3]. Disfluency counts are often done in real-time by the SLP as a client is talking. However, these are not very specific, as SLPs might only count the total number of disfluency rather than count each type separately. Blocks, prolongations, and sound repetitions are often viewed as more indicative of stuttering than phrase repetitions, revisions, and interjections. The disfluency counts can also not easily be re-examined, without relistening to the entire audio, as disfluencies counts are not tied to timepoints in the audio file.

An alternate to real-time counts is the verbatim transcript approach. Here, the SLP first transcribes exactly what was said, and then marks up the transcript with disfluency codes [4, 5, 6]. This allows the SLP can review their annotations. Furthermore, the type of each disfluency is identified, as well as what words they occur on, allowing for a much richer analysis of a client's disfluency patterns. However, few tools exist to help with this type of disfluency counts. Automatic approaches to transcribe stuttered speech and count disfluency have been attempted (e.g., [7, 8]), but Automatic Speech Recognizers (ASRs) have problems even with fluent speech. Due to the amount of time that this approach can take, this approach is rarely used by SLPs in clinical practice, and only sometimes used in research studies.

Our long-term goal is to build computer tools that help SLPs and researchers count disfluencies that can be used to analyze a client's disfluency patterns. Rather than striving for a fully automatic tool, which would transcribe and annotate stuttered speech, we have set a more modest goal of *assisting* the SLP in this process. Our goal is reduce the time required of an SLP to produce verbatim transcript disfluency counts from over 20 times real-time to just several times real-time.

In previous work [9], we proposed a multi-step process for creating the transcript-based disfluency counts, given below. For that work (and the current work), we focus on read speech, where the client is reading a story out loud. Such speech samples are regularly used in stuttering diagnosis and treatment [10] and allow us to leverage the highly constrained nature of the speech to have usable ASR output, even for stuttered speech.

Step 1: The SLP annotates the disfluencies in real-time, perhaps as the client is speaking, using 8 categories: sound, word and phrase repetitions, revisions, interjections, blocks, prolongations, and other [2, 11].
Step 2: An ASR uses the text from the story to produce a word transcription (possibly with errors in it).
Step 3: A computer program merges the ASR transcript with the SLP's annotations to produce an annotated verbatim transcript. Each of the SLP's annotations are placed on a word in the transcription that it is likely to have occurred on.
Step 4: A computer program determines a set of regions for the SLP to review and correct.
Step 5: The SLP reviews and corrects the word transcription and disfluency annotation for each region.

With this approach, a relatively complete *annotated verbatim transcript* can be produced with a single real-time pass and a set of samples that are re-heard and corrected, if need be.

In our previous approach, the SLP's annotations are applied in Step 4 after the ASR produces the 1-best word transcription in Step 3. So, the SLP's annotations are not leveraged in finding the best word transcription. Thus, if the SLP annotated a word repetition, hypotheses with a word repetition in the vicinity would not be given preferential treatment. In this paper, we use an ASR to create a word lattice. We then search through the lattice to find an optimal path, making use of the acoustic and language model scores, and the clinician's real-time annotations.

This study was funded by the NIH under grant 2R42DC009944-02 to BioSpeech, where Drs. Heeman and Lunsford were employees. This potential conflict of interest has been reviewed and managed by OHSU.

http://dx.doi.org/10.21437/Interspeech.2016-1388

In the rest of the paper, we first describe our development and test data in Section 2. In Section 3, we analyze the clinicians' annotations to understand how they might be used to improve ASR. Section 4 describes how we create the lattices using Sphinx4. Section 5 describes how we augment our algorithm for finding paths through the lattice to take into account the clinicians' real-time annotations. Section 6 gives an evaluation of our results. In Section 7, we conclude and discuss the relevancy of this work to other annotation tasks. Any time that a clinician is classifying events that are occurring in a client's speech could potentially be used in a rescoring pass to improve the transcript.

2. Development and Testing Data

All of the data used in this study is based on audio files of children, aged 8 through 12, who stutter, reading aloud short stories. The audio files are between 1 and 2 minutes in length. These files were transcribed using SpeechView [12]. We also annotated these files with a disfluency annotation scheme [13] that is more fine-grained than the 8 annotation codes that clinicians typically use. These annotations were done very carefully (and very slowly), but by someone without clinical experience.

As we are interested in aligning *real-time* clinician annotations, we have created a development and test set. First, we had one of the co-authors, who is an SLP, use prototypes of our real-time annotation tool to annotate some of the audio files that we collected. This data is only used in the development of our annotation alignment tool, and refer to it as **DevA**.

The second set of data was collected in a formal evaluation of our annotation tool and our review and correction tool. In that study, we compared how well SLPs annotated disfluencies with our computer tools versus pencil-and-paper using the 8 categories of disfluencies. We recruited five SLPs who regularly see clients who stutter and regularly use a computer. They received 4 hours of training, of which about one third was about applying the disfluency scheme in real-time with paper and the computer tools.

Subjects did two sessions, each lasting about 1.5 hours, separated by one week. Subjects annotated 20 audio files, from our corpus of stuttered speech, which were not used in **DevA**. We grouped the audio files by speaker, and included a practice audio file at the beginning of each group, so as to familiarize the SLPs with the speaker's disfluency patterns. With each audio file, SLPs alternated using pencil-and-paper to count the number and type of disfluencies, and using the annotation and correction tools (Step 1 and 5). Half of the subjects used paper first, and the rest used the computer tools first. For the second session, subjects switched which method they used for each file.

In all, each clinician annotated all 20 files with the annotation tool, giving us time-aligned annotations. For each clinician, we put 8 files into a development set, referred to as **DevB**, and 12 files into a test set, alternating which files are in each set for each clinician.

3. Quality of Real-time Annotations

In order to make use of clinician real-time annotations in improving the ASR output, we need to understand how good these annotations will be. As the SLP will be annotating disfluencies in real-time, as the client is speaking, there is no time to review them and correct any mistakes. Clinicians will sometimes miss a disfluency, annotate one by mistake, or annotate it with the wrong type. Due to human reaction time [14], the annotation will not be exactly where the disfluency occurred, but it should be within a couple of seconds.

		F	I	Rv	Rp	Rw	Rs	P	B	O
F	Fluent		1	0	3	4	6	13	8	0
I	Interjection	47	**0**	1	0	0	2	0	3	0
Rv	Revision	30	0	**33**	11	7	0	5	1	1
Rp	Phrase Rep	22	0	3	**33**	6	0	0	0	1
Rw	Word Rep	26	0	0	4	**55**	7	2	0	2
Rs	Sound Rep	69	0	1	0	15	**58**	7	10	0
P	Prolongation	115	0	0	1	1	4	**61**	4	0
B	Block	31	0	1	0	0	5	6	**22**	0
O	Other	0	0	0	0	0	0	0	0	**0**

Table 1: Confusion Matrix

First, we compare the real-time clinician annotations to the reference annotations. As the reference annotations are in a different scheme, we wrote a computer program to convert them, and then used dynamic programming to best align the reference and clinician annotations that were in a 5s window, as we did in our earlier work [9]. We checked the output of the algorithm to make sure it was making reasonable alignments.

In Table 1, we give a confusion matrix of the reference annotations (rows) versus the clinician real-time annotations (columns) for the development set **DevB**. The bold entries along the diagonal shows where the two annotations agree. The column labeled 'F' shows the number of annotations that the clinician missed. The row showing 'F' shows the annotations where the clinician annotated a disfluency where there is not one in the reference annotation. The other entries show disagreements between what was annotated. There are some patterns in terms of which annotations tend to be easily confused with other annotations. For example, sound and word repetitions are confusable, as are blocks and sound repetitions.

Second, we compute the average lag time for the clinicians' real-time annotations that were correctly aligned with a reference annotation for **DevB**. For the reference annotation, we use the time that the disfluency could have been noticed. Consider a phrase repetition, say 'we can go to . can go to the store'. In the reference annotation, the phrase repetition code of 'Rp' would be placed on the first instance of 'to'. However, it is only after the speaker says the second instance of 'can' that a clinician has any hope of identifying it. So, for phrase repetitions, we compare clinician's time against the end of the word 'can'.

Type	Number	Average Lag	SD
Rv	33	2.32s	0.80s
Rp	33	2.01s	0.85s
Rw	55	1.74s	0.81s
Rs	58	1.73s	0.93s
P	61	1.39s	1.01s
B	22	2.05s	1.07s
All	262	1.79s	0.96s

Table 2: Lag times

Table 2 shows the average lag times. As can be seen, sound and word repetitions are annotated faster than revisions and phrase repetitions. This might be because more context than the first word of the new material might be needed to identify them. These results on confusion and lag times will influence how we incorporate the clinicians' real-time annotations in our processing.

4. Creating the Word Lattice

To incorporate clinicians' real-time annotations, as well as more complex modeling of disfluency patterns, we rescore the word lattice from an ASR. Our audio files range between 1 and 3 minutes in length. For this project, we are using Sphinx 4 [15], which has problems producing word lattices for audio files longer than 12 second. Hence, we segment the audio files into smaller files, run the ASR on each to produce the word lattice for each, and then *stitch* them together so that our rescorer can run across the entire file.

There has been a large amount of work done on speech segmentation (e.g., [16]). However, our domain of read speech spoken by someone who stutters requires special consideration. First, the story text constrains what the speaker will say, which makes a decoder-guided approach practical, in which an ASR is run on the entire file to find silence regions. We run Sphinx 4 on the entire audiofile to produce a 1-best word transcription. As described in earlier work [9], we hand-crafted a bigram language model based on the story being read and common word patterns for disfluencies: a speaker tends to say the next word, say just part of the next word, say an interjection, or backtrack to an earlier place in the story (typically the beginning of a phrase, or the current sentence).

Typically with the decoded-guided approach, the output is used to locate silence regions, at which the audio file is segmented. The second issue that we must consider is that we are dealing with stuttered speech. Disfluencies in stuttered speech, especially multi-iteration sound repetitions, prolongations, and blocks, pose difficulties for ASRs, and might be misrecognized as silences. Hence, for stuttered speech, silences are not an ideal place to segment an audio file, as we want the lattices to capture other alternatives for the silence region.

We want to segment the audio file at places where the one-best output of the ASR is most likely to be correct. First, people rarely stutter on the last word of a sentence, and so the last word of a sentence is likely to be correctly recognized by the ASR. Using the ASR output matched to the story text, we can identify all of the sentence endings; these are candidates for segmenting the audio file. Second, people who stutter will be fluent for stretches of speech. Hence, we look for fluent word sequences (where words are being said in the exact order of the story and without any pauses). The middle of these sequences are candidates for segmenting the audio file.

The candidates for segmentation are also given a score. For cutting in the middle of a sequence, the longer the fluent stretch on both sides of the cutpoint, the higher the rank. For end-of-sentence candidates, ones that are followed by a silence are scored higher, as well as ones that have a fluent stretch before them. We then use a greedy algorithm to find a set of segmentation points which gives as few segments as possible, and where the segments are no longer than 12s.

Once we have the segmentation, we run Sphinx 4 on the smaller audio files. For the language model, we do not use the entire story text, but the part of the story that was identified in the 1-best word transcription of the audio segment. The language model used in this second pass is a richer language model that also accounts for sequences of interjections.

Finally, we stitch together the word lattices from each segment. Each of the individual lattices has a start and an end node. We simply change all of the arcs going to the end node in one lattice to instead go to all of the nodes that follow from the start node.

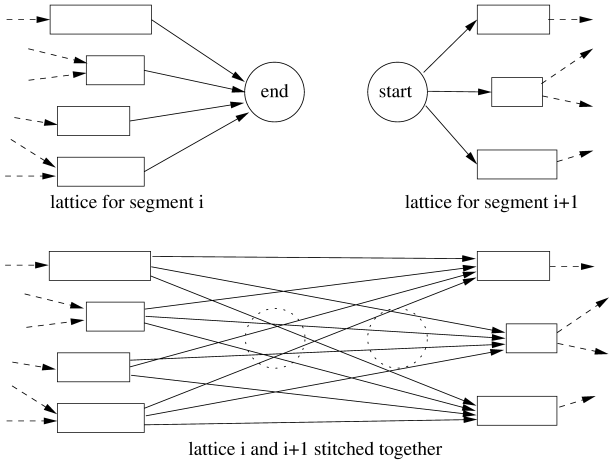

Figure 1: Stitching together lattices

5. Rescoring the Lattice

We rescore this lattice with our language model, to account for word transitions across the segment boundaries.

We rescore with a path-based language model that can deal with longer distance dependencies, which phrase repetitions and revisions require [17]. This is also required by multi-iteration word and sound repetitions, where the same word or sound is said several times in succession.

For our evaluation, we compare the output of the speech recognizer and rescoring passes against the hand-transcribed reference (Section 2). We use word error rate (WER):

$$WER = \frac{substitution + insert + misses}{correct + substitutions + misses}$$

The results from each pass for our test set are given in Table 3. As can be seen, after segmenting the audio files into smaller segments, and rescoring with the language model, we get a small decrease in performance, from 7.27% to 7.55%. Rescoring with our richer language model brings us back to 7.27%. As this model is data-driven, we expect that as we collect, transcribe and annotate more data, that this result will improve.

	First-Pass	Lattices	Rescore	Oracle
Total	7.27%	7.55%	7.27%	2.62%

Table 3: Results of Producing a Lattice

For reference, we also show the WER of the best possible path through the lattice (using an oracle). This has a WER of 2.62%. About half of the error in the oracle rate is due to words that are not in the language model, for example when a speaker inserts the word 'the' or 'a' where there is not one in the story, or uses an interjection that we did not model, such as 'like' or 'I mean'. The purpose of our work is to use the clinicians' real-time annotations to improve the WER from 7.27% to closer to 2.62%.

6. Rescoring with Annotations

We use the algorithm from the previous section that searches though paths through the lattice. We currently focus on revisions, interjections, and phrase, word and sound repetitions. In future work, we will incorporate blocks and prolongations.

Path Structure: For each partial path that is being considered, we keep track of which annotations have yet to be used. For each word in the path, we keep track of which clinician annotations have been assigned to it. For interjections, this is the interjection word. For repetitions, this is the last word of the material that is about to be repeated. For multi-iteration word and sound repetitions, this is the the first occurrence.

Algorithm: We augment the code from the previous section, as shown below. When extending a path p for a word w, we first determine how relevant each possible annotation is (including fluent), given the path p and the current word w. For each of the SLP's annotations that are in the vicinity and that have not yet been aligned (including no annotation), we create a new path, using the language model (LM) probability of the word w given the path p and the factor for that annotation.

for each time point t
  for each path p that ends at time t
    for each word w that can extend path p
      compute new LM prob for w given p
      from p+w, determine how good any annotation is
      for each SLP annotation (including none)
        not aligned yet?
        in right time frame?
        How well does it match the word+context
          adjust LM score
          create new path

What annotations make sense from the transcription? For a path p and next word w, we determine what annotation tags make sense. This task is simplified as we know the index of each word in p+w into the story. Hence, if the index of the current word precedes the index of the previous word, then the speaker is likely backtracking, and making a phrase repetition or a revision. If the index is the same, the speaker is likely making a sound or word repetition.

Rather than determining a single annotation tag that makes sense, we evaluate how likely each tag is. In this way, we can model how the SLP might confuse one annotation for another. Furthermore, we include fluent (F) as one of the options so we can account for whether the clinician missed it. Note that even when no disfluency annotation makes sense given the transcript, we still have a score for each annotation. The scores for each annotation are used to either reward or penalize a word transcription (described later in this section).

In accounting for the time lag, we take into account what word a disfluency can be noticed by, versus which word in the transcription should have the annotation mark (see description of Table 2). As we are processing the current word, we will look for disfluency patterns in which the current word is involved. For example, for phrase repetitions, say 'we can go to . can go to the store', for each word of the repetition, 'can', 'go' 'to', we will consider this as a target for a phrase repetition annotation tag from the clinician, so long as we haven't already assigned the first instance of 'to' with a disfluency tag.

Penalizing and rewarding paths through the lattice. Each possible tag has a factor assigned to this. Due to the small amount of development data, we tuned these factors by hand to optimize performance on the **DevA** and **DevB**. These factors either penalize or reward a word transcription. We are trying out the clinician annotations over a time-window. But, each clinician annotation must be used once. So, if there is a word sequence that is consistent with that annotation somewhere in the time-window, that path will be rewarded. If there is no consistent word sequence in the path, the path will be penalized.

How much to penalize the time lag? As shown in Table 2, clinician annotations tend to lag by a certain amount. Hence, we modify the factors to take into account how far they are from the ideal time.

7. Evaluation

As discussed in Section 2, our test data consists of 12 audio files for each of the 5 clinicians. Table 4 gives the results of our rescoring with the clinicians' real-time annotations, in the column labeled 'Annotations'. For the reader's convenience, we repeat the four columns of results from Table 3. Overall, on the test set, we improve the WER from 7.27% to 6.92%. This is a relative improvement of 4.8%. Given that the best possible path through the lattice has an error rate of 2.62%, the relative improvement with respect to what is possible is 7.5%.

We also show the results for each clinician. We see that for each of the 5 clinicians, the WER on their 12 audio files improved, and that it improved by at least 0.18% absolute.

	First-Pass	Lattices	Rescore	Annotations	Oracle
C1	5.49%	5.91%	5.73%	5.55%	1.73%
C2	6.80%	7.04%	7.00%	6.80%	2.68%
C3	7.05%	7.48%	7.27%	6.94%	2.42%
C4	7.82%	7.95%	7.61%	7.18%	2.97%
C5	8.87%	9.14%	8.64%	8.10%	3.21%
Total	7.27%	7.55%	7.27%	6.92%	2.62%

Table 4: WER Results

8. Conclusion

In this paper, we presented our work on using speech technology to build better tools for clinicians to use in diagnosing and treating people who stutter. Using the audio file of a person reading a text, along with a clinician's time-aligned disfluency annotations, we use an ASR and lattice rescoring to produce an *annotated verbatim transcript*. We find that we can improve the quality of the word transcription by incorporating the clinician's annotations by a relative factor of 7.5%. This will mean clinicians will need to spend less time correcting the word transcription, and can spend more time reviewing the client's stuttering patterns, and thus personalizing the therapy for client. We expect that as we collect more data, we can employ data driven techniques to further improve the transcription.

This work is not only relevant for stuttering. Tests in which a trained clinician monitors a speaker while "scoring" the speaker's response are a standard part of the batteries used to assess cognitive and speech skills. One example is the Weschler Logical Memory test, used to assess immediate and delayed memory recall [18]. For this test, the speaker is asked to retell a story the clinician has read to them, both immediately after hearing the story and again after 15 minutes of other tasks. During retelling, the clinician is expected to create an abbreviated transcript of the story retelling, while also noting which of 25 story elements the speaker has recalled – a challenging real-time annotation task. However, using techniques such as those described herein, the clinician might need to only identify the recalled story elements, and rely on ASR to produce the transcript and count the number of unique recalled elements. Furthermore, the resulting time-aligned transcription could be used to measure other features of the speech, such as syntactic complexity, pause-rate, and hesitation, that are also indicative of cognitive impairment [19, 20].

9. References

[1] J. S. Yaruss, "Clinical measurement of stuttering behaviors," *Contemporary Issues in Communication Science and Disorders*, vol. 24, pp. 33–44, 1997.

[2] ——, "Real-time analysis of speech fluency: Procedures and reliability training," *American Journal for Speech-Language Pathology*, vol. 7, no. 2, pp. 25–37, 1998.

[3] E. G. Conture, *Stuttering: its nature, diagnosis, and treatment*. Allyn & Bacon, 2001.

[4] N. Bernstein Ratner, B. Rooney, and B. MacWhinney, "Analysis of stuttering using CHILDES and CLAN," *Clinical Linguistics and Phonetics*, vol. 10, no. 3, pp. 169–187, 1996.

[5] H. H. Gregory, J. H. Campbell, C. B. Gregory, and D. G. Hill, *Stuttering Therapy: Rationale and Procedures*. Pearson Allyn & Bacon, 2003.

[6] J. Campbell, D. Hill, and M. Driscoll, "Systematic Disfluency Analysis: Using SDA to determine stuttering severity," in *Annual Convention of the American Speech-Language-Hearing Association*, Anaheim, CA, Nov. 1991.

[7] P. Howell, S. Sackin, and K. Glenn, "Development of a two-stage procedure for the automatic recognition of dysfluencies in the speech of children who stutter: II. ANN recognition of repetitions and prolongations with supplied word segment markers," *Journal of Speech, Language, and Hearing Research*, 1997.

[8] E. Nöth, H. Niemann, T. Haderlein, M. Decher, U. Eysholdt, F. Rosanowski, and T. Wittenberg, "Automatic stuttering recognition using hidden markov models," in *Proceedings of the 6th International Conference on Spoken Language Processing (ICSLP-00)*, Beijing, Oct. 2000.

[9] P. A. Heeman, A. McMillin, and J. S. Yaruss, "Computer-assisted disfluency counts for stuttered speech," in *Proceedings of the 12th Annual Conference of the International Speech Communication Association*, Florence Italy, Aug. 2011, pp. 1324–1327.

[10] G. Riley, *Stuttering Severity Instrument*, 4th ed., 2009.

[11] J. S. Yaruss, M. Max, R. Newman, and J. Campbell, "Comparing real-time and transcript-based techniques for measuring stuttering," *Journal of Fluency Disorders*, vol. 23, pp. 137–151, 1998.

[12] S. Sutton, R. Cole, J. de Villiers, J. Schalkwyk, P. Vermeulen, M. Macon, Y. Yan, E. Kaiser, R. Rundle, K. Shobaki, P. Hosom, A. Kain, J. Wouters, M. Massaro, and M. Cohen, "Universal speech tools: the CSLU toolkit," in *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP-98)*, Sydney Australia, November 1998, pp. 3221–3224.

[13] P. A. Heeman, A. McMillin, and J. S. Yaruss, "An annotation scheme for complex disfluencies," in *Proceedings of the 9th International Conference on Spoken Language Processing (ICSLP-06)*, Pittsburgh PA, Sep. 2006, pp. 1081–1084.

[14] D. B. Fry, "Simple reaction-times to speech and non-speech stimuli," *Cortex*, vol. 11, no. 4, pp. 355–360, 1975.

[15] P. Lamere, P. Kwok, E. Gouvea, B. Raj, R. Singh, W. Walker, M. Warmuth, and P. Wolf, "The CMU sphinx-4 speech recognition system," in *Proceedings of the IEEE International Conference on on Acoustics, Speech and Signal Processing*, Hong Kong, 2003.

[16] D. Rybach, C. Gollan, R. Schülter, and H. Ney, "Audio segmentation for speech recognition using segment features," in *ICASSP*, 2009, pp. 4197–4200.

[17] P. A. Heeman, "Modeling speech repairs and intonational phrasing to improve speech recognition," in *Automatic Speech Recognition and Understanding Workshop*, Keystone Colorado, December 1999. [Online]. Available: http://www.cse.ogi.edu/ heeman/papers/97.asru.html

[18] D. Wechsler, *Wechsler Memory Scale - Third Edition Manual*, San Antonio, TX, 1997.

[19] B. Roark, M. Mitchell, J.-p. Hosom, K. Hollingshead, and J. Kaye, "Spoken language derived measures for detecting mild cognitive impairment," in *IEEE Transactions on Audio, Speech & Language Processing*, 2011.

[20] R. Lunsford and P. A. Heeman, "Using linguistic indicators of difficulty to identify mild cognitive impairment," in *Proceedings of the 17th Annual Conference of the International Speech Communication Association*, Dresden Germany, Sep. 2015.