Detection and Recognition of Stutter in English Speaking Adults Using Automated Speech Recognition

Carolyn Holz

MIT 6.345

cjholz@mit.edu

Abstract

Disphony Disorders such as stuttering can be a major obstacle in a person's life, especially if they do not receive adequate treatment and the stutter continues into adulthood. Approximately 5% of people develop a stutter between the ages of two and six and 20% of them will carry their stutter into adolescence or even adulthood[1]. A speech disorder in adulthood can be an obstacle to many aspects of a person's life including interaction with technology. Many of our devices require or at least offer a voice command feature. People with speech disorders are excluded from full use of these products.

This work improves speech recognition for stuttered speech by first implementing stutter detection. A stutter can then be classified within unseen speech and removed before speech recognition. We classify segments of speech as having or not having a stutter based on a collection of free form and structured utterances recorded for adult speakers who stutter.

Index Terms: speech disorders, human-computer interaction, speech classification

1. Introduction

Speech Disorders are a fairly common occurrence in the population, yet we do not adequately accommodate this type of disability in our technology. This is particularly troubling because technology could be an extremely useful tool in combating this issue. Very often speech disorders can be detected and treated in childhood so that they do not persist into adulthood. However, many factors can prevent this from happening because speech therapy is a time intensive and expensive process. If we could provide a technology that detected and assisted in speech therapy exercises at home some of this burden could be alleviated and many speech disorders such as stuttering could be improved well before adulthood.

Stuttering manifests in three major types. A repetition stutter is a sound or word repeated, usually at the beginning of a word or utterance. A prolongation occurs anywhere in speech and is characterized by a sound being stretched out for longer than normal. The hardest type of stutter to predict is a blocking stutter, when someone has trouble beginning to make a sound at all. This is particularly difficult because this type of stutter often sounds like silence. Because the range of possible stutters is so variable, the task of recognizing them automatically in speech is particularly challenging.

We separate the task of improving recognition of stuttered speech into two distinct tasks, classification and recognition. The first task classifies speech as normal or as a specific type of stutter in order to detect where a stutter within a speech file. Once we have detected the segments of audio where the stutter occurs, we begin our second task. We can remove the detected stutter segments and what remains will be normal speech. A speech recognition model should then be able to more accurately transcribe this cleaned audio.

2. Related Work

2.1. ASR for Aphasia

ASR has been used as a therapy tool for other speech disorders as described in Jamal et al [7]. Aphasia is disorder characterized by the inability to convert thoughts into speech or written words. ASR could be useful here both to prompt the subject for their next word and to convert speech to text.

2.2. Classifying Stuttered Speech

A similar approach to our own is described in Chopra et al [6]. They experiment with many traditional ASR features but approach the problem of classifying speech as stutter or not stuttered in a segmented way. Instead of attempting to recognize the speech they merely classify small pieces of it. We will take a similar approach with the intent to clean the speech to improve recognizer performance.

3. Dataset and Preparation

3.1. Data Description

Our abnormal speech dataset consists of 34 interviews and 23 reading exercises done by 40 different adults. The speaking tasks are in English and include free response to interview questions and structured reading. Each exercise includes a video file and rich transcript including clinician annotations and start and end times for each utterance corresponding to the video file.[2]. The recordings include utterances from the interviewer but both tasks consist primarily of speech by stutterers. In total, by length of utterance, the dataset is 89% utterances by stutterers. The approximate distribution of speaking time for subjects and interviewers separated by gender is shown in Table **??**.

3.2. Preprocessing

We preprocessed our data separately for our two tasks. The first task requires uniform length audio segments with labels indicating the classification of the audio as stutter or non stutter. The second task requires more traditional data structure in the form of utterances and their corresponding text transcription.

3.3. File Conversion

The original data files are in mp4 video format and CHAT transcription format which are not formats that are traditionally used in ASR. We converted the files to more traditional formats before continuing with preprocessing. We converted the video files to wav files using ffmpeg. We converted the CHAT transcription files to XML using the Chatter software provided by

talkbank [**?**] for later preprocessing.

3.4. Labelling

We hand labelled ten reading files and one interview file with our six classes of audio: silence, normal speech, repetition stutter, prolongation, blocking stutter, and noise. The files were labelled to a granularity of one tenth of a second. We used Serato DJ Pro [9] to play the audio which provides playback times up to a tenth of a second granularity. We were able to view the waveforms in the application and select the moment that speech began which allowed to be extremely accurate in our manual labeling. We attempted to label the same files with start and end times of utterances as well as stutter tags at the appropriate places in the transcription but this proved to be prohibitively time consuming.

3.5. Uniform Splitting of Audio

For our first task we split the audio files we had manually labelled into uniform segments labelled with the class that occurred most often within them. We used pydub [8] to split the original audio files into segments of one tenth of a second, one half second, one second, one and a half seconds, and two seconds. We then used our manual labelling to count the number of each class of label within the interval and selected the class that occurred most frequently.

3.6. XML Parsing

Our original CHAT files were parsed into XML which contained information about utterances including each word within the utterances, some stutter labelling, and start and end times for each utterance as well as whether the utterance was from the interview or participant. The XML also included some information about the file such as interviewer and participant gender. We were able to parse the XML to create text for each utterance, however many stutter tags and start and end times were missing. We attempted to label some of the files ourselves before deciding to take an approximate approach. Instead of finding the exact start and end time for each utterance we relied on the available start and end times and segmented the audio based on those times. In this way, we combined utterances with missing start and end times using the end time of the last known utterance and the start time of the next utterance found.

3.7. Google API

Approximating start and end times meant that we could not rely on the text utterances found in the XML matching the audio segments we created. Instead we used Google's Speech to Text API [10] to create transcription for our approximate audio. In doing so we excluded utterances which returned no transcription and which were over the maximum limit of one minute. A total of 168 utterances were excluded, 161 returned no transcript and seven were too long for transcription leaving us with 2744 utterances in total. We extracted both the transcription and confidence from the API to use as an evaluation metric for our detector.

3.8. Stutter Removal

After training our model we predict stutters and other non speech in the rest of our dataset and remove those segments from our utterances. We split each utterance created in the XML parsing step into the proper length segment for the model and compute are features for them. We then concatenate the normal speech segments to produce cleaned speech and re-transcribe the speech using Google's API.

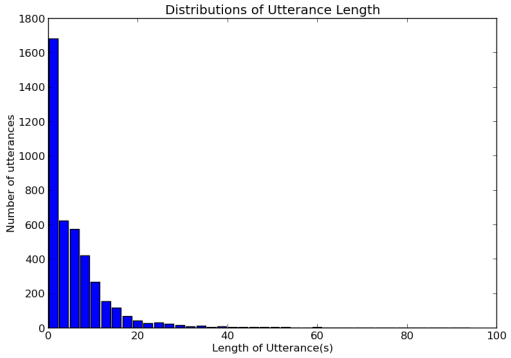

Figure 1: *Histogram of Utterance Length*

Table 1: *Classification distribution across subjects [2]*

Speaker Type	Gender	Percentage
Interviewer	U	4.9%
	M	2.8%
	F	3.3%
Subject	U	22.5%
	M	47.9%
	F	18.6%

4. Methods

We split our method into two tasks. The first is to detect and classify segments of speech as stuttered or not stuttered. In addition, we attempt to classify the type of stutter into one of the three identified categories. Our second task is to remove segments of speech that are not classified as normal speech and recognize the remaining audio with a traditional recognizer.

4.1. Classification

We define six classes of audio as described in section 3.4. We hypothesize that classifying not only a stutter but the specific type of stutter will improve our performance because the audio signals for different types of stutters may be vastly different. In addition classification brings us closer to the goal of a therapeutic tool for stuttering, since knowing which stutter type is most prevalent in a patient will help suggest appropriate therapies.

This step is done with a simple two layer neural network. We chose a simple architecture in the interest of time and because a more complicated architecture did not provide better results on preliminary analysis. The Network consists of a convolutional layer followed by a max pooling layer and an output dense layer with six output nodes. These are the probabilities of each class and we select the highest probability as our output label. We compute audio features including MFCC, chroma, mel spectrogram, tonal centroid features and spectral contrast.

4.2. Recognition

Due to missingness in our data and time constraints we use Google's Speech to Text API to perform this second step. We request transcriptions from the API for each utterance and observe the confidence for each as well as the average confidence for all utterances. This allows up to evaluate the performance of our detector in removing stuttered speech. We expect the average confidence to improve after stutters have been removed.

5. Experiments & Results

5.1. Classification

For classification we determine our best model by experimenting with different audio segment lengths, parameters and signal features as described in [6]. We define three objectives to optimize, our validation loss, our validation accuracy on all classifications, and our accuracy in detecting the binary label of stutter or non-stutter. Using an initial audio segment length of one second we tuned six model parameters: number of epochs, batch size, dropout rate, learning rate, activation and optimizer. We first trained our model on only the labelled reading files to determine if classification was feasible. Since the reading files contain nearly identical utterances we assumed that if we were not able to perform well on this data we would not be successful on the less uniform interview files. Once we added our interview segments we tuned our parameters again with similar results as shown in Table 2. Once we had tuned our parameters we tested different features both on the reading data and after including interview data. Each model is trained on MFCC features with the addition of test features including chroma, mel spectrogram, tonal centroid features and spectral contrast. The feature selection results are shown in Table 3.

In both the reading only model and the model trained using mixed readings and interviews we discovered similar parameters and useful features. We set our parameters and features accordingly and experimented with different uniform audio segment lengths. The results of this experiment are shown in Table 4. We discovered that the optimal segment length was our most granular of one tenth of second. After adjusting our batch size to 100 to accommodate our higher number of samples our final model validation loss was .694 with a test accuracy of 75.89% and binary stutter prediction accuracy of 86.18%.

5.2. Recognition

Our recognition experiments are a validation step to evaluate the accuracy of our stutter detection. We produce transcripts and confidence scores from Google's API three times, once on uncleaned speech and once on speech cleaned using our most accurate detection labelling. We compare Google's average confidence over all utterances to determine the success of our detector. In our first experiment we produced fairly accurate transcriptions as shown in Figure 2, with an average confidence score of 84.1%. In our second experiment, transcribing audio cleaned manually we achieved a confidence score of 95.3%. In our final experiment, transcribing audio cleaned by our model we produced very accurate transcriptions with an average confidence score of 91.53%.

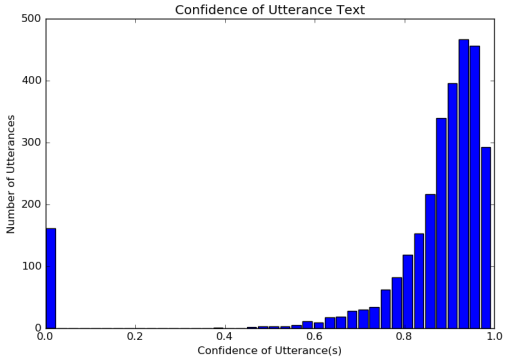

Figure 2: *Histogram of Confidence for Original Utterances*

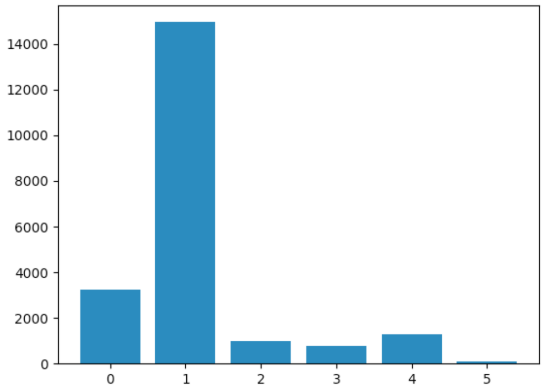

Figure 3: *Histogram of Labels for Uniform Audio*

6. Discussion

6.1. Classification

We were surprised by the parameters we discovered for our classification model. Given that our shortest observed stutters were a third of a second in length we expected that longer uniform audio segments would be needed to classify stuttered speech. We hypothesize that some of the benefit of using the shorter length came from the fact that we had many more samples to work with. We had 21323 one tenth of a second samples. At any of the longer audio lengths we would have had an order of magnitude fewer samples. We were also surprised by the accuracy of the stutter detection across all classes. Given the fact that stutter types can often be mixed in a single word we had expected 15 to 20 percent accuracy boost between exact classifications and binary classifications. In addition, we expected that the mismatch in our samples, skewed toward normal speech would make it harder to classify a specific stutter type. We hypothesize that the uniformity of the reading data may have been beneficial in counteracting this.

6.2. Recognition

We were impressed with the confidence score produced by the speech cleaned using our model. We expected it to be closer to the uncleaned speech given our 86% accuracy in classifying stutters. It is not surprising that it was not as accurate as manually cleaned speech since the manually cleaned speech was

Table 2: *Best Parameter for Data Types*

Data Type	Epochs	Batch Size	Dropout Rate	Learning Rate	Optimizer	Activation	Validation Loss
Reading	50	50	.5	.001	Adam	Softmax	.69697
All Files	50	20	.5	.001	Adam	Softmax	.9294

Table 3: *Analysis for Feature Selection*

Data	Features	Validation Loss	Validation Accuracy	Binary Label Accuracy
5*Reading	MFCC only	.85114	74.54%	81.25%
	MFCC + Spectral Contrast	.81977	75.28%	83.1%
	MFCC + Tonal Centroid	.82106	75.28%	83.8%
	MFCC + Mel Spectrogram	.94599	71.22%	80.88%
	MFCC + Chroma	.87171	73.06	79.78%
5*All Files	MFCC only	1.0302	66.9%	75.82%
	MFCC + Spectral Contrast	1.02995	64.32%	74.88%
	MFCC + Tonal Centroid	1.0125	67.61%	75.35%
	MFCC + Mel Spectrogram	1.081	65.73%	75.35%
	MFCC + Chroma	1.03297	67.37%	74.18%

Table 4: *Analysis for Audio Segment Length*

Segment Length	Validation Loss	Validation Accuracy	Binary Label Accuracy
.1 seconds	.69697	76.03%	85.83%
.5 seconds	.9294	68.57%	79.7%
1 second	1.0225	65.73%	74.88%
1.5 seconds	1.1757	55.12%	74.56%
2 seconds	1.0908	58.96%	72.17%

done using very accurate audio playing software. We did expect higher confidence with our manually cleaned speech but the lower confidence may have been a result of noise in the recording since the recordings were done in a small room with a recorder set on the table.

7. Future Work

7.1. Labeling

Time constraints meant that we could only label a subset of our data ourselves. Given more time we would have at least three licensed therapists label the data. This would give us more sizable and accurate labelling and improve our performance. In addition labelling would be done both for detection and recognition. For detection, our labelling process would remain as describe in Section 3.2. For recognition, the therapists would tag stutters as a separate token within the transcribed text such that the stutter would be treated as a word. This would allow us to train a recognizer which would include stutter tokens in it's output and would be more usable in a live speech recognition model. A therapist would also be able identify stutters that are a combination of more than one stutter type which could improve our accuracy and sensitivity.

7.2. Modeling

We would like to experiment with more complex neural networks as well as other classification algorithms. We think that clustering maybe well suited to this task since our problem consists of well defined classes.

7.3. Recognition

We have stored the transcriptions obtained for each type of utterance: stuttered, cleaned using manual labels and cleaned using automatic labels in order to train a recognizer on the output. We expect that there will now be some missing speech where normal speech was removed and would like to experiment with ways to handle this with a recognizer. In addition we would like to train a recognizer that would treat stutters as a token and label them in the text output.

8. Acknowledgements

We would like to thank the 6.345 staff for their help in focusing the scope of the project as well as providing feedback on this two step approach. We would also like to thank Nan Berenstein Ratner and FluencyBank for the use of this dataset.

9. References

[1] "Stuttering Facts and Information" Stuttering Foundation of America.

[2] N. Ratner, "Voices of Adults Who Stutter" FluencyBank. Fileset, 2017.

[3] "Chatter", https://talkbank.org/software/chatter.html TalkBank. Software, 2019.

[4] NIDCD, "Specific Language Impairment" National Institute on Deafness and Other Communication Disorders, 2017.

[5] M. Pützer, "Saarbruecken Voice Database" Institut für Phonetik Universität des Saarlandes.

[6] M. Chopra, "Classification and Recognition of Stuttered Speech" Stanford.

[7] N. Jamal, AIP Conference Proceedings 1883, 020028 (2017); https://doi.org/10.1063/1.5002046 Published Online: 14 September 2017

[8] pydub https://github.com/jiaaro/pydub GitHub, 2019.

[9] Serato DJ Pro https://serato.com/dj/pro Serato, 2019.

[10] Google Cloud Speech to Text https://cloud.google.com/speech-to-text/ Google, 2019.