Segment-Removal Based Stuttered Speech Remediation

Pierre Arbajian[1], Ayman Hajja[2], Zbigniew W. Raś[1,3,4], and Alicja A. Wieczorkowska[3]

[1] University of North Carolina, Dept. of Computer Science, 9201 University City Blvd., Charlotte, NC 28223, USA
[2] College of Charleston, Dept. of Computer Science, 66 George Street, Charleston, SC 29424 USA
[3] Polish-Japanese Academy of Information Technology, Koszykowa 86, 02-008 Warsaw, Poland
[4] Warsaw University of Technology, Institute of Computer Science, Nowowiejska 15/19, 00-665 Warsaw, Poland

`arbajian@uncc.edu`
`hajjaa@cofc.edu`
`ras@uncc.edu`
`alicja@poljap.edu.pl`

Abstract. Speech remediation by identifying those segments which take away from the substance of the speech content can be performed by correctly identifying portions of speech which can be deleted without diminishing from the speech quality, but rather improving the speech. Speech remediation is especially important when the speech is disfluent as in the case of stuttered speech. In this paper, we describe a stuttered speech remediation approach based on the identification of those segments of speech which when removed would enhance speech understandability in terms of both speech content and speech flow. The approach we adopted consists of first identifying and extracting speech segments that have weak significance due to their low relative intensity, then classifying the segments that should be removed. We trained several classifiers using a large set of inherent and derived features extracted from the audio segments for the purpose of automatic improvement of stuttered speech by providing a second layer filtering stage. This second layer would discern the audio segments that need to be eliminated from the ones that do not. The resulting speech is then compared to the manually-labeled "gold standard" optimal speech. The quality comparisons of the resulting enhanced speeches and their manually-labeled counterparts were favorable and the corresponding tabulated results are presented below. To further enhance the quality of the classifiers we adopted a voting techniques that encompassed an extended set of models from 14 algorithms and presented the classifier performance measures from different voting threshold values. This voting approach allowed us to improve the specificity of the classification by reducing the false positive classifications at the expense on additional false negatives thus improving the practical effectiveness of the system.

Keywords: stuttering detection, speech analysis, speech remediation, classification

1 Introduction and Background

Quality of life is negatively impacted when individuals are chronically unable to express themselves due to lack of speech fluency. Stuttering, also referred to as stammering, is a condition that is exhibited in bad fluency of speech. The onset of stuttering generally occurs during childhood years, and in one fourth of cases this speech impediment persists throughout life [8, 11].

Automatic speech disfluency detection offers many advantages, ranging from time saving, constant speech monitoring, reducing the subjectivity of manual disfluency identification [9, 5] as well as a more effective automatic speech recognition. Therefore, the identification of "episodes" of stutter will offer firm and actionable information [3–5]. Reliable identification of speech segments, from the beginning of episode to the end of episode, with pauses, blocks, interjections and hesitations will allow speech cleanup by ridding the speech of the blocks, and interjections, smoothing the hesitation segments and shortening the prolongations [6, 7, 10].

The state of the art is rich with papers which describe various prosodic- and semantic-based machine learning approaches and algorithms for the detection of disfluent speech [1, 9, 11–13]; however, research work addressing disfluent speech remediation is considerably less common. Stuttered speech is difficult to listen to; our assumption in this research work is that speech clarity will be significantly improved by eliminating those segments. As a result the listener will more intent on listening to the speech.

To confirm the assumption that undesired speech segment removal enhances speech quality, we ran a script that eliminates audio segments with low intensity levels using varying intensity thresholds and segment durations. This resulted in an unquestionably clearer and more intelligible audio compared to the original speech. That being said, automatically detecting and removing these *potentially undesired* segments, resulted occasionally in discarding the wrong segments. For that reason, we introduced a second layer of filtering stage when the *potentially undesired* segments are further classified into *true undesired*, which would need to be discarded; and *false undesired*, which would need to be retained.

The complexity in this paper arose from the need to take into consideration various overlapping multi-threshold data samples. Also, we devised a labeling process which reduces the effort required during the labeling process. We feel that the proposed adaptive data collection and processing method paves the way and lays a sound foundation for successful treatment of future and similar multi-perspective [15] data collection and classification challenges.

In this paper, we discuss one certain type of stuttered speech remediation; remediation in the form of undesired speech segments removal. In the next section, we present an algorithm to enhance the stuttered speech quality by eliminating the speech segments that contain stuttering *blocks*, which are defined as the segments [14] of audio in which the person who stutters is unable to produce clear sounds. Furthermore, unwanted segment elimination provides the listener with yet another benefit; that is, the speech is shortened (up to 60% reduction) without speeding up individual syllables.

Eliminating disfluent segments of speech requires identifying those segments which do not add any value to the speaker's message. Our algorithm first identifies and creates sound files of those segments that have the distinct potential of being undesired in the speech through linguistic intensity and length analysis; detailed explanation will be provided in Section 2. Next, the set of *potentially undesired* audio segments are listened to in order to determine whether they have semantic value or not, and to determine whether they must be deleted or retained accordingly. The final output of the speech audio file, after removing the segments that were manually-labeled *true undesired*, will represent the "gold standard". The "gold standard" speech is later used to measure the performance of the trained classifier. There are two vital reasons for the phase in which the *potentially undesired* segments are manually labeled; the first reason is to generate the optimal speech that will be used to evaluate our enhanced stuttered speeches, and the second reason is to generate the *true undesired* and *false undesired* segments used to build our classifier.

The dataset of speeches that we used was obtained from the UCLASS series of recordings. UCLASS, which stands for the University College London Archive of Stuttered Speech, is a database of stuttered speech recordings with individual speech and speaker metadata. UCLASS contains a collection of over one hundred speeches from which we chose fifteen for our research study.

The approach that we followed was to begin by scanning each speech for potentially stuttered segments according to varying intensity thresholds and lengths of audio segments. Every *potentially undesired* extracted segment is considered a candidate for removal because of the selection criteria used; however, in numerous cases, the candidate segment may contain semantic speech and therefore should not be omitted from the original speech. Determining which candidate segments are truly undesired candidates and which segments are not, is done through a classification system that was trained by providing it with manually-labeled *true undesired* and *false undesired* segments that were collected beforehand.

The features that are used in the classifier training include data extracted from both the frequency domain such as voice formants and pitch, and from the time domain such as intensity. In addition to that, a combination of speaker and speech metadata are also used to improve the trained classifier; a more elaborate description of the list of features will be presented in the next section.

The difference between the quality of the "gold standard" of a given speech compared to the quality of an enhanced speech after having every *potentially undesired* segment being classified is used to evaluate the quality of our system. The results with respect to classifier performance are tabulated in the findings section according to the extraction parameters. Process efficiency improvements were created to minimize manual labeling effort as well as segment deletion redundancy.

As we remove speech portions we must be careful to avoid deleting good content i.e. we are quite reluctant to remove segments carrying meaning. We consider such mistakes more detrimental to the speech than not removing blocked voice portions. In order to minimize the rate of mistaken speech segment removal we doubled the number of classifiers and performed the classification according to a threshold vote cutoff. Instead of a majority vote we selected multiple voting proportion cutoffs, tabulated and graphed the outcome for the respective True Positives, True Negatives, False Positives, False Negatives, Accuracy, Specificity, and True Positive Rates. This exercise was instrumental in helping us decide on a voting threshold which greatly reduces the number of wrong segment omissions at a reasonable rate of increase in non removal of blocked speech segments.

2 Proposed Method

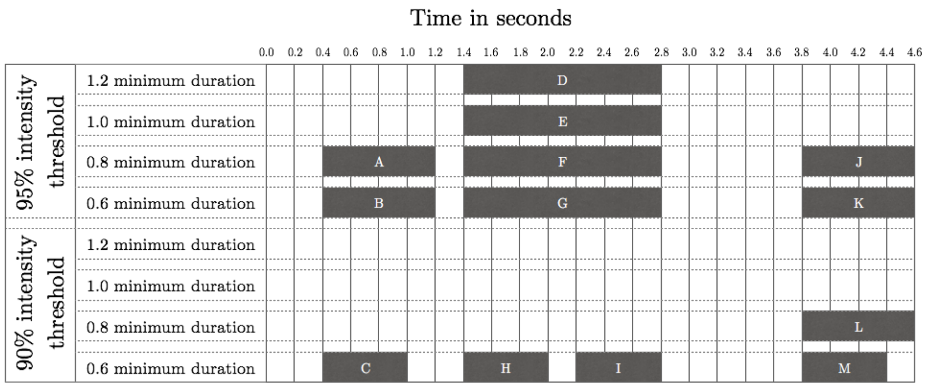
The method we adopted for this research consisted of selecting a set of speeches to be examined from a group of disfluent speeches available from UCLASS. UCLASS speeches exhibit a wide variety of speakers and stuttering conditions, yet the vast majority of the speeches are distinctly stuttered and quite disfluent. We selected fifteen speeches (roughly one hour of stuttered speech) as the foundation for our research. Our goal in this research is to develop a system capable of automatically analyzing stuttered speeches for the purpose of detecting undesired segments and enhancing the overall speech quality through eliminating the disfluent segments. In this work, we used a *Praat* script for initially detecting potential candidate segment to be removed from the stuttered speech; the ultimate goal of this research is to build a system able to distinguish *true undesired* from *false undesired* for all candidate segments. Next, we provide an elaborate description of the workings of the overall process.

2.1 Extracting Potentially Undesired Candidate Segments

Fifteen speeches from the UCLASS repository are selected as the stuttered speeches of interest, each is scanned with a total of 8 segment extraction parameters. The extraction parameters determine what the sound intensity threshold and the minimum duration of those segments should be. The minimum durations that we used to iterate through our speeches are 0.6, 0.8, 1.0 and 1.2 seconds, and the speech intensity thresholds are at 95% and 90% of the entire speech intensity in decibels; for example, if the average intensity of a given speech is 50 decibels, then the intensity thresholds that correspond to 95% and 90% are 47.5 db and 45 db, respectively.

For each possible pair combination of segment duration and intensity threshold (total of 8 unique combinations), the entire speech is scanned and the *potentially undesired* candidates are detected according to the corresponding pair combination. The minimum segment duration ensures that no excessively short segments are extracted as potential candidates for deletion, which would disrupt the flow of normal speech and unfavorably affect the speech cadence. Note that the extracted segments from each unique pair will not overlap, and that only such overlapping is possible, and rather likely to happen, when examining segments extracted from different pairs. We provide a demonstration of how the segments may look like in Figure 1. For example, according to Figure 1, extracting potentially undesired segments from the fourth row (0.6 minimum duration with 95% intensity threshold) will result in three different segments (B, G, and K); since segment B's length is 0.8 seconds, which satisfies the minimum duration threshold for the third row, then the same segment will also be identified as a *potentially undesired* segment on the third row; however, since the length of the segment is not 1.0 second (or more), that segment will not be identified on the second (or first) row in Figure 1.

Fig. 1. Extracting potentially undesired candidate segments



Each extracted candidate segment results in a waveform audio file accompanied by five additional vectors that serve as the segment inherent features, three out of which contain the values of formant 1, formant 2, and formant 3 and the remaining two vectors contain the pitch values and the intensity values. All formants were computed with a 50 ms Gaussian analysis window with a 12.5 ms offset. The pitch values were taken at 10 ms interval and computed with the *Praat* software algorithm which performs an acoustic periodicity detection on the basis of an accurate autocorrelation method as described in [2]. The intensity values were computed by squaring then convolving the sound values within Gaussian analysis window of 50 ms length. The waveform audio files will be later used for manually labeling the candidate segment, and the set of inherent features (in addition to a set of derived features presented in Subsection 2.3) will be used to build our classifier.

2.2 Labeling Potentially Undesired Segments

Following the generation of the segment files, the labeling process is performed to begin building the classifier and testing our dataset. The segment candidates must be manually labeled as: 1) delete and analyze, these are denoted by G, 2) leave in the speech and analyze, these are denoted by B, and 3) no analysis, but delete from the speech, these segments of speech represent external sounds such as when the interviewer was attempting to speak softly. We chose to exclude these segments from the speech during cleanup because they showed no speech disfluency but refrained from including them in the training dataset.

The "delete and analyze" labels (G) indicate that the sound is void of semantic meaning and that we would like for our classifier to mark such segments as "delete", or as *true undesired.* The "leave in the speech and analyze" label is used for sounds that were long enough and with low intensity yet the segment sound contained semantic meaning and must not be removed from the original speech; our classifier should label such segment as "retain", or *false undesired.* The "delete and do not analyze" label indicates that a segment is best deleted from the speech but does not represent the type of sound that our classifier must learn to recognize or take any action about; an common example would be a segment containing the soft voice of the interviewer.

In an attempt to avoid redundant labeling efforts and minimize the number of segments which must be manually reviewed, we have sorted the list of segments to be reviewed such that if a segment s is marked "delete", then all other segments that are contained within s (start with, or after, s; and end when s ends, or anytime before that) will also be marked as "delete". The reasoning behind our approach is that if some segment s is void of semantic meaning, then any other segment contained in s must also be void of semantic meaning. Note here that it is not the case that the opposite is true; given that some segment s contains some semantic meaning, we cannot conclude that all other segments contained in s must also contain semantic meaning. This approach has allowed us to reduce review time by a factor of five.

We will use Figure 1 to demonstrate the process explained above. The first step is to sort all *potentially undesired* segments according to their start time, from top to bottom. This means that based on Figure 1, the list of candidate segments will be sorted as follows: A, B, C, D, E, F, G, H, I, J, K, L, and M. The first candidate segment to be listened to is A; if the segment A is marked as *true undesired* (no semantic value), then all other subsets that are contained in A are also marked *true undesired*, which are B and C. Similarly, if segment D is labeled as *true undesired*, then segments E, F, G, H, and I are also labeled *true undesired.* If however, segment D was marked as *false undesired* (contains semantic value), then this only implies that segments E, F, and G are also *false undesired* (since they are essentially the same as segment D), but we cannot conclude that segment H nor I are also *false undesired*; therefore, we would need to listen to the two segments H and I to determine what the label of each of them should be.

The dataset used in our classification exhibited two possible class values: G (positive), which meant that the candidate sound must be deleted; and B (negative), which meant that the sound segment should not be deleted.

2.3 Building the Classifiers

In addition to the label feature, the dataset used to train our classifiers will contain two additional types of features: 1) a set of derived attributes extrapolated from the set inherent features, and 2) speaker and speech metadata [15]. The inherent features that are associated with each candidate segment are formant 1, formant 2, and formant 3 (sampled at 50 ms intervals with the time step being 12.5 ms), pitch (sampled at 10 ms), and intensity (sampled at 8 ms). The derived features that are used in our classifier training are all extrapolated from the inherent features and will serve as a set of data-points that provide information about the entire segment as opposed to an exact point in time (due to sampling). We start by calculating the derivatives of each one of the five inherent features; the reason for calculating the derivatives is to assess the variance of our inherent features, which is vital for detecting stutter. Then, for each one of the inherent features and their derivatives, we calculate the average, median, standard deviation, percentiles (25, 50, and 75), minimum value, maximum value, peak-to-peak amplitude measurement, and variance.

The speaker and speech metadata we included in our dataset consisted of those data features provided with the UCLASS dataset; they consisted of the following:

Speaker category metadata: Gender (M/F), Handedness (L, R, not known), Past history of stuttering in the family, Age of stuttering onset, Age at the time of recording, Location of recording (clinic, UCL, or home), Recording conditions (quiet room or sound-treated room), Type of therapy received (family based treatment or holistic treatment), Time between therapy and recording time, Speaker had any history of hearing problems (Y/N), Speaker had a history of language problems (Y/N), and Special educational needs (Y/N).

Speech category metadata: Background acoustic noise level (numeric), environmental noise level (numeric), speaker clarity (numeric), interlocutor intrusiveness (numeric).

The resulting dataset consisting of signal statistics, metadata, and labels consisted of 126 features and was used to train eight different classifiers, each for a unique pair combination of minimum duration and intensity threshold (see Algorithm 1). Recall that the minimum duration threshold is the minimum length of *potentially undesired* segments, and that the intensity (percentage) threshold is measured by examining the averaged speech intensity. We used the *R* 'caret'package in order to streamline the classifier testing process and cover a wide range of classifiers.

Algorithm 1: Summary of our proposed algorithm

```
for every segment duration and intensity threshold pair do
    extract potentially undesired candidate segments from a given speech;
    manually label each candidate as true undesired or false undesired;
end
training and testing phase;
for every segment duration and intensity threshold pair do
    train a classifier using a portion of the labeled candidate segments;
    test the remaining segments using the classifier built above;
    evaluate the resulting labels by comparing them to the "gold
      standard";
end
```

There are three different courses of action for segments removal that are examined in this research work, each producing a different level of stutter remediation:

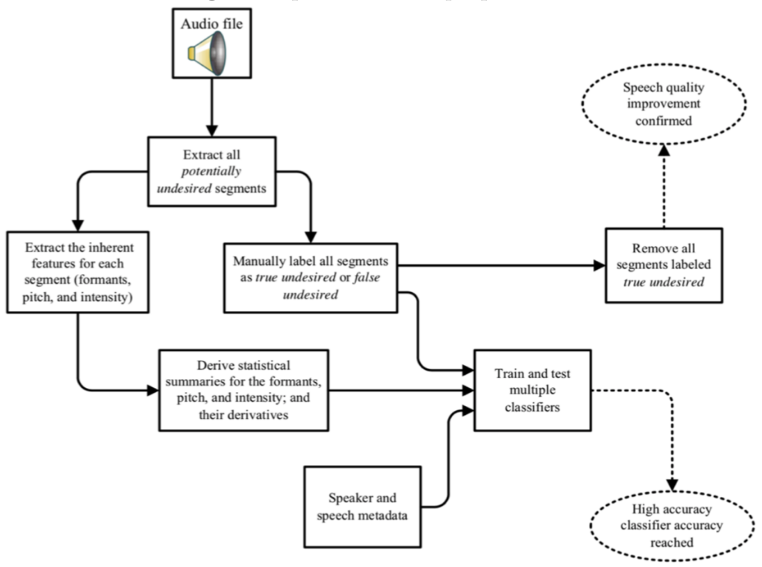
1. Remove all candidate segments without human intervention (this approach does not require manual labeling).
2. Remove only the candidate segments that are manually labeled *true undesired* (resulting speech is referred to by the "gold standard").
3. Classify all candidate segments using our trained classifier, and only remove the candidates that are classified *true undesired.*

Because of the speech intensity weakness during the episodes to be deleted we were able to remove the segment minus 50 ms from the front both the front and end of the episode without creating any perceptible sound discontinuities.

Each removal course of action yields a certain "enhanced" speech file; the enhanced files are then examined and compared. Our goal is to build a classifier that generates an audio file (course of action number 3) that is as close as possible to the golden standard audio file (generated from course of action number 2).

In the case of speech remediation, it is more important to avoid deleting semantically meaningful segments than to miss deleting semantically meaningless segments. In other terms, when evaluating our classifier performance, false negatives (deleting what should not be deleted) should be considered more undesirable than false positives (missing to delete segments which must be deleted). The *R* classifiers that we trained and tested are C5.0, Neural Networks, Recursive Partitioning, Random Forests, Polynomial SVM, and AdaBoost. In Figure 2 below we show the process flow starting at the top left *Extraction* of the candidate segments followed by two independent paths (1)*Feature* extraction and (2)*Labeling* ; the results of the two flows provides the digital signal processing predictors and manual label results which will be combined with the *metadata* to create the complete dataset.

Fig. 2. Depiction of our proposed method


Audio file
Extract all *potentially undesired* segments
Extract the inherent features for each segment (formants, pitch, and intensity)
Manually label all segments as *true undesired* or *false undesired*
Remove all segments labeled *true undesired*
Speech quality improvement confirmed
Derive statistical summaries for the formants, pitch, and intensity; and their derivatives
Train and test multiple classifiers
Speaker and speech metadata
High accuracy classifier accuracy reached

2.4 Practical classifier enhancement

We mentioned the impact of removing *block* episodes throughout the paper and established the detrimental effect of removing episodes which are not truly bad. In this section, we will revisit the segment removal concept as it pertains to the optimization of speech repair.

The benefits of the identification and removal of *blocked* segments is multi-fold; the listener benefits in a multitude of ways:

1. He/she does not hear spurious sounds which provide no meaningful lexical content to the recording.
2. The flow of the recording is conducive to a smoother listening experience because of increased fluency.
3. The speech is shortened thus reducing speech length.

The speaker benefits also, we list two such advantages:

1. By listening to the repaired speech, he/she acquires the confidence that some prosodic speech repairs will greatly improve the message delivery.
2. The speaker can listen to the message on the merit of its meaning instead of style thereby focusing on the core semantic components of the recording

Original, pre-repair speeches are qualitatively weak, yet their prosodic and lexical content is uncompromised. If sought indiscriminately, the benefits listed above could compromise the speech where meaning loss is incurred due to the repair process.

The aim of the described system enhancement is to generate a speech which is superior in practical usability without the collateral damage of lexical component resulting from removing segments that were not supposed to be removed.

The reason for remedy optimization is to ensure that as little meaning loss as possible is caused thanks to the intervention from our proposed process. To this end, we remain cognizant that meaning loss is more "detrimental" to the message conveyed by a recording than the inconvenience of hearing unnecessary *blocks* of speech. By more detrimental we mean that we would prefer to retain stutter speech episodes if eliminating them means losing other lexical content. Therefore, we will explore the options available to us which help with the remediation process without cognitive meaning compromise. The level of tolerance to collateral lexical damage incurred from our remediation approach and attempts to improve a message prosodic quality must be carefully considered.

In the following remedy improvement discussion, we will use classifier prediction *confusion matrix* results and derived measurements to help frame the proposed solution qualitatively, and quantitatively.

		Prediction outcome	
		p	**n**
actual value	**p′**	True Positive	False Negative
	n′	False Positive	True Negative

Table 1. Confusion Matrix

The confusion matrix Table 1 provides general measurements which apply to classification prediction models. We will describe the confusion matrix and measurements as they relate to our proposed remediation environment in general and the improvement covered in this section.

In Table 1 the prediction outcome corresponds to the segment evaluation in our classification, column *p* represents a segment evaluation of *positive*; a *positive* evaluation means that a sound episode which was extracted as a candidate for removal is evaluated as a *blocked* utterance segment and must, according to our classifier prediction, be removed from the speech.

The *n* column in the Table 1 confusion matrix represents a candidate speech episode which was presented as candidate for deletion, but the classifier prediction evaluated it to be a segment with lexical content and therefore, the classifier indicates that it must be retained in the recording. The negative (n) column will be used to denote candidate speech episodes which were presented as candidates for deletion, but the classifier predictions evaluated them to be segments with lexical content and therefore, they should be retained in the recording. So, p means that a classifier suggests a segment must be removed and n indicates that a classifier judged a segment as *negative for removal* and thus it must remain in the recording. The *actual value* rows consist of the results obtained from the manual segment labeling effort. Manual labeling is assumed to provide accurate results, and *True* or *False* according to that assumption.

The p' row represents a speech extracted candidate segment which, when manually reviewed, was labeled as a *blocked* stutter segment and contained no lexical meaning. If we had a perfect candidate extraction system, all p' segments would be predicted as such by our classifier(s) and consequently removed.

The n' row represents speech candidates presented for removal consideration but when *manually* reviewed they were labeled as lexically meaningful and therefore should ideally be excluded from the list segments to remove from the speech.

The matrix cells are represented by four squares: *True Positive*, *False Positive*, *False Negative*, *True Negative* at the intersection of the *actuals* and *prediction.*

True Positive represents the number of candidate segments that were manually reviewed and determined to be of no value to the speech and evaluated by the trained classifier model as "bad", lexically void, segments. This is a case where both the *actual/manual* accurate values and the classifier *predicted* values agree.

True Negative represents the number of speech segments manually labeled as lexically meaningful hence they should be retained; and, consistently, these segments were classified as lexically meaningful by the trained model. This, also, is a *prediction* outcome that agrees with the *actual* value of the segment.

The *True Positive* and *True Negative* cells represent the desired outcome and pose no concern; if all our samples fell in these two categories we would consider our classifier to be performing in an absolutely optimal way.

A *False Negative*, means that the segment is manually labeled devoid of lexical meaning (e.g. "Positive") but the trained classifier was not able to detect that and as a result it assigned it a class of *Negative*, thus a disfluent segment, and it will be left in the recording and no repair will be performed.

The *False Positive* classifications occur when the manual, accurate, label indicates that the candidate segment is a *negative* segment implying that the segment contains semantic meaning, yet the classifier indicates that the segment should be deleted. The *False Positive* cells count represents the total number of samples that fell in that category at time of prediction. These *False* classifications cause a loss of speech semantic content.

The *False Negative* and *False Positive* cells in the confusion matrix represent predictions for which we want to insightfully fine tune our model classification capabilities, thereby improving the practical usability of our classifier.

False Positive conditions lead to speech content compromise due to removal of speech parts which contain semantic information. *False Positive* mistakes are fundamentally detrimental to the message and should be avoided if possible, albeit at a price. Avoiding a False Positive means that we will incur additional *False Negatives*; it is a trade off because the classifier is generally tuned to minimize the total number of misclassification thus maximizing accuracy. One can assume that by further tuning the classifier to minimize the total number of False Positives would result in adding proportionately more *False Negatives.*

Deciding on our level of tolerance for lexical loss is of central importance to the repair we propose. How many bad episodes are we willing to leave in a recording to avoid losing a good segment is an essential question to the work body of this research. Namely, how many false negatives would we be willing incur to avoid one *False Positive.*

To realize our tolerance for *False Positives*, we consider the False Positive Rate (FPR), False Negative Rate (FNR) and *Accuracy*. False Positive Rate (FPR) represents the percentage of *Negative* samples that were classified positively relatively to the total number of *Negative* samples.

The lower our FPR is, the less mistakes our classifier would have made in classifying segments for removal when those should have been retained in the speech. Therefore, we would like the FPR to be as low as possible; and since we would like to compare our measures to the accuracy which ranges between 0 and 100%, we will consider Specificity (SPC) instead of FPR ($SPC = 1 - FPR$). When working with SPC we would aim to maximize SPC to a value as close to 100% as possible.

Also we will consider the False Negative Rate (FNR). FNR is the percentage of samples classified as negative when the actual class is *Positive*; these are segments that must be deleted, however our classifier has classified them for retention. These mistakes are not as damaging to the speech and will have a lesser priority. We would be willing to trade multiple *False Negatives* for one *False Positive.* For ease of comparison with *Accuracy* and $Specificity$, we will be considering the True Positive Rate ($TPR = 1 - FNR$).

There are multiple ways to perform classifications that tilt the balance in favor of more *False Negatives* than *False Positives* (or the opposite).

1. *Cost based training:* Train individual classifiers by placing a higher penalty on *False Positives* (*FP*) than *False Negatives* (*FN*) thus creating models which are less prone to *FP*s.
2. *Sampling technique:* During the classification inflate the number of positive samples in the training sets by using same Positive samples multiple times, thereby resulting in a higher count of Positive samples. This approach, also, creates classifier models with higher sensitivity to *FP* than to *FN* classifications.

3. *Probability cutoff:* During training adjust the probability cutoff for classifier models to favor minimizing *False Positives* as opposed to being optimized to minimize both FNs and FPs combined.

4. *Voting consensus:* One could also rely on a voting approach where one would take pre-trained classifiers and use a consensus among multiple classifiers to determine a final class.

Because we had many tuned and trained classifiers ready for use, we chose the fourth approach and experimented with a voting mechanism that utilizes all classifier results to consider collectively, and reach an optimal FPR. We chose to devise a voting mechanism that utilizes the result of 14 classifiers to extract a subset which, when combined, provided all we needed to perform the next steps.

In this case, we chose to vary a percentage of votes threshold as the method to decide whether a segment should be classified Positive or Negative, i.e. *Delete* or *Retain* a segment. In preparation of the dataset, we averaged the results of all classifiers and varied the decision cutoff threshold to classify every sample as Positive or Negative.

For this experimentation, we used the models that were trained and tuned within the 'caret'package with a balanced dataset described in prior sections. The results from the models are combined with the actual label which consists of 15 columns; one actual class (e.g. the manual label) and the 14 classifiers results.

We used the resulting table of 15 columns to evaluate the impact of changing the voting threshold on the $Specificity$ (SPC), *True Positive Rate* (TPR) and *Accuracy*. As previously stated, because false positives are especially detrimental to the remediation process, we wanted to maximize SPC while maintaining acceptable *Accuracy* and TPR levels. To make FPs less frequent, we intuitively expected that the higher the majority vote requirement we impose on the *votes-based* classifier, the less FPs (e.g. High SPC) we will end up with.

As shown in Table 2 we varied the voting threshold (Θ) to range from 14/14 to 1/14, where Θ denotes the number of classifiers needed to classify a segment s as *Positive*. This means that if Θ is set to 14/14 (leftmost row in Table 2), the audio segment will be classified as *Positive* if all 14 classifiers vote *Positive*; if Θ is set to 13/14 the audio segment will be classified as *Positive* when 13 or more classifiers vote *Positive* etc..

In addition to *True Positive*, *True Negative*, *False Positive*, and *False Negative* totals, we measured the *Accuracy* rate, $Specificity$, and TPR corresponding to each of the 14 voting threshold results. As can be shown by Table 2, our initial intuition that to minimize FPs we must use higher threshold was confirmed.

Table 2. TP, TN, FP, FN with Specificity, TPR, Accuracy

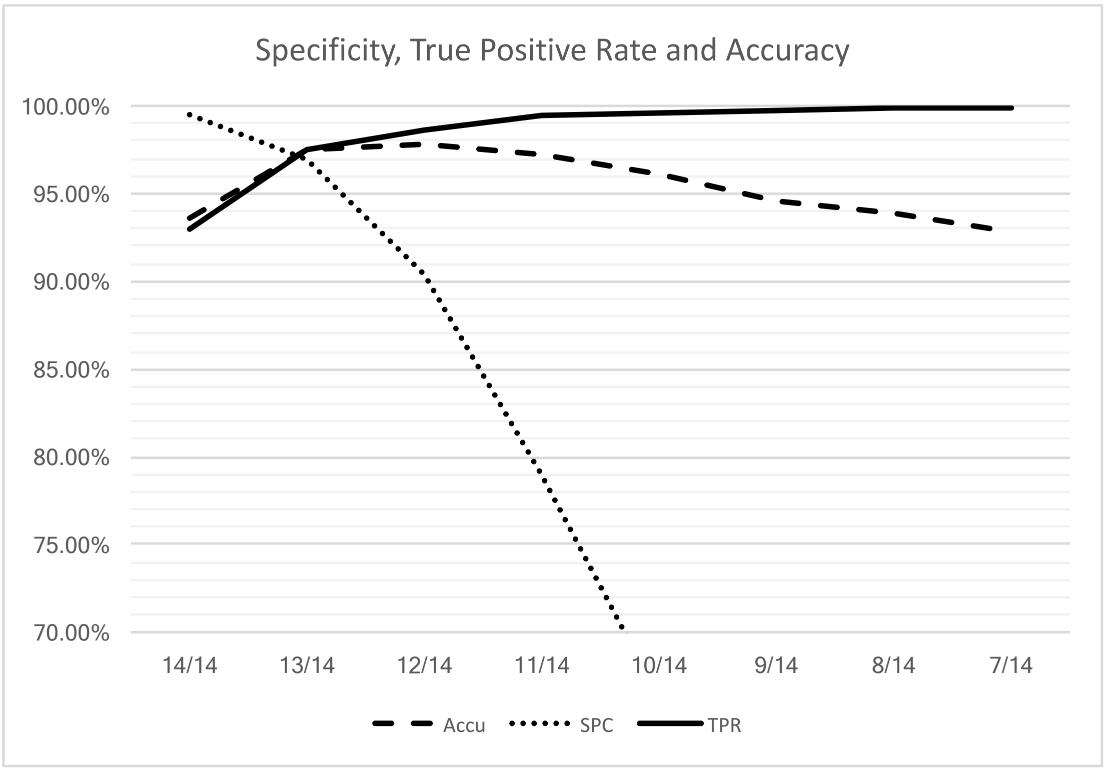
						1-FPR	1-FNR
Vote ϴ	**TP**	**TN**	**FP**	**FN**	**Accu**	**SPC**	**TPR**
14/14	1601	198	1	121	93.65%	99.50%	92.97%
13/14	1679	193	6	43	97.45%	96.98%	97.50%
12/14	1699	180	19	23	97.81%	90.45%	98.66%
11/14	1712	157	42	10	97.29%	78.89%	99.42%
10/14	1715	132	67	7	96.15%	66.33%	99.59%
9/14	1717	99	100	5	94.53%	49.75%	99.71%
8/14	1720	84	115	2	93.91%	42.21%	99.88%
7/14	1720	63	136	2	92.82%	31.66%	99.88%
6/14	1721	50	149	1	92.19%	25.13%	99.94%
5/14	1722	40	159	0	91.72%	20.10%	100.00%
4/14	1722	33	166	0	91.36%	16.58%	100.00%
3/14	1722	23	176	0	90.84%	11.56%	100.00%
2/14	1722	0	199	0	89.64%	0.00%	100.00%
1/14	1722	0	199	0	89.64%	0.00%	100.00%

The analysis of Table 2 leads to the observation that there is a reverse correlation between FP and FN. A close review of *Accuracy* (*Accu*), SPC and TPR shows the best Accuracy results (97.81%) to occur at $\Theta = 12/14$ but the nature of our speech remediation prefers lower FP classes even if there is a relatively larger increase in FN classifications. By considering $\Theta = 13/14$ we lower the FP values from 19 to 6, namely, we end up with 13 less instances where we remove an episode which contains actual meaning; while increasing the FN classifications from 23 to 43 thereby increasing the number of stutter *blocks* that remain in the speech by 20. So, we choose to trade 13 FPs for 20 FNs and left 20 *blocks* in the recording.

Such a trade off is reasonable when considering the high negative impact False Positives have on a recording and we find that varying a voting cut off threshold brings considerable improvement to the overall practicality of our system.

The effect of moving the Θ threshold is further illustrated in Figure 3 where the line graph depicted over a narrow range of Θ (14/14 to 7/14) and a narrow range of $Specificity$, TPR and *Accuracy* (70% to 100%) visually magnifies the impact of Θ and helps us confirm the soundness of our decision to use a Θ value of 13/14.

Fig. 3. Specificity, TPR, Accuracy (Accu) according the voting threshold



3 Experiments and Results

Our approach proved to be effective in eliminating the vast majority of voice blocks when applied to stuttered speech. The results of disfluent speech remediation consisted of the elimination of anomalous speech and a reduction in speech length between 60% for the most severe stuttered speech to about 25% for mildly stuttered speeches.

The effectiveness of our remediation process is highly dependent on the threshold of the speech intensity tolerance during the potentially undesired segments identification. As we have stated earlier, we used two different thresholds for the sound intensity for the initial phase - not to be confused with the voting threshold. On the one hand, the lower threshold (90% of average) resulted in a considerably less undesired candidates as shown in Table 3; although most of the undesired candidates extracted using the lower threshold were actually true undesired, there were many instances where stutter segments were not detected (due to the low threshold). On the other hand, the higher threshold (95%) resulted in a much higher number of potentially undesired segments, some of which were false undesired; however, most of the stuttered segments were detected in the first phase. Therefore, the value of the second phase (training and testing phase), which identifies true undesired and false undesired from potentially undesired segments, is most useful when the threshold is high, and when most of the actual stutter segments are detected, even if that means potentially marking some segments as potentially undesired while in reality they are false undesired, during the first phase.

We have subsequently added a component to this system of remediation which helped us to enhance the usability of the system. Because a good remediation process should lose little to no semantic meaning we implemented a *False Positives* restricting subsystem which relies on a voting cutoff approach. This enhancement would only tag a segment for removal if the vote for removal is almost unanimous amongst all qualifiers. Although our accuracy was negatively affected by this component, the results of this experimentation proved to be effective in reducing *False Positives.*

Table 3. Segment count

	90% of speech avg sound volume threshold	95% of speech avg sound volume threshold
0.6 seconds	195	211
0.8 seconds	149	159
1.0 seconds	114	130
1.2 seconds	83	93

The second threshold value that we used during the extraction of the potentially undesired sound segments was the minimum time duration, starting with 0.6 seconds and ending with 1.2 seconds. It is worth noting that we tried shorter duration time frames with little success, causing the comprehensibility of the speech to be diminished; we found 0.6 second minimum silence duration to be the lowest value that can be used in our experiments.

We used two different approaches for treating our data with each classifier. The first approach is to build one classifier for our data after balancing the labels regardless of the minimum duration and intensity threshold values; in other words, we combine all the segments extracted from all eight different unique pairs of minimum duration and intensity threshold, and build our classifier accordingly. The balancing approach we chose to apply for the balancing of the data consisted of randomly excluding data tuples of the over-represented class G.

The second approach is to build a single classifier for each unique pair, then average the accuracy and confusion matrix results. Table 4 shows the results obtained using the first approach, while Table 5 shows the results obtained from using the second approach for the two classifiers that performed the best using separate classifiers for every unique pair; Random Forests and C5.0.

Using 10-fold cross validation, we trained six different classifiers: C5.0, Neural Networks, Recursive Partitioning, Random Forests, Polynomial SVM, and AdaBoost. During the training of our classifiers, we used the 'caret'R package because of its built-in tuning functionality. The parameters to build the respective models were chosen based on highest accuracy. We have listed the 'caret'package chosen parameters below:

Neural Network (nnet): Tuned model parameters when training on the balanced dataset: size = 5 and decay = 0.1. Tuned model parameters when training on the entire dataset: size = 1 and decay = 0.1. Size represents the number of units in the hidden layer and can be zero if there are skip-layer units. The decay parameter represents weight decay.

Random Forest (rf): Tuned model parameters when training on the balanced dataset: mtry = 2. Tuned model parameters when training on the entire dataset: mtry = 56. The parameter mtry represents the number of variables randomly sampled as candidates at each split.

SVM Polynomial (svmPoly): Tuned model parameters when training on the balanced dataset: degree = 3, scale = 0.01 and C = 1. Tuned model parameters when training on the entire dataset: degree = 3, scale = 0.01 and C = 1. The degree parameter represents the polynomial degree of the kernel function. The scale is the scaling parameter of the polynomial and tangent kernel. C is the cost regularization parameter.

Adaptive Boosting (AdaBag): Tuned model parameters when training on the balanced dataset: mfinal = 100 and maxdepth = 3. Tuned model parameters when training on the entire dataset: mfinal = 150 and maxdepth = 3. The mfinal parameters represents the number of iterations for which boosting is run or the number of trees to use. Maxdepth is the maximum depth of any node of the final tree.

Recursive Partitioning (rpart): Tuned model parameters when training on the balanced dataset: : cp = 0.03797468. Tuned model parameters when training on the entire dataset: cp = 0.05970149. Cp is the complexity parameter; any split that does not decrease the overall lack of fit by a factor of cp is not attempted. The main role of this parameter is to save computing time by pruning off splits that are evidently not worthwhile.

As can be seen in Table 4, when we balance our data we reduce the number of samples in the dataset, this big reduction in the number of tuples seems to have a detrimental impact on our neural network classifier model, since neural network training is best accomplished with large datasets.

Table 4. Results obtained from building a single classifier using all of segments

	True Positive Rate	True Negative Rate	Accuracy
C5.0	100%	94%	97.4%
Neural Networks	75.6%	55.9%	67.7%
Recursive Partitioning	92.6%	94.2%	93.2%
Random Forests	98.4%	91.7%	95.8%
Polynomial SVM	89.9%	75.6%	84.4%
AdaBoost	98.3%	93.8%	96.4%

Table 5. Results obtained from using a single classifier for each pair of minimum duration and intensity threshold, and averaging the results

	Random Forest	C5.0 Classifier
True Positive Rate (Averaged)	95.3 %	94.8%
True Negative Rate (Averaged)	82.3 %	76.6%
Classifier with Highest True Positive Rate	1.2 minimum duration, 90% intensity threshold	0.8 minimum duration, 90% intensity threshold
Classifier with Highest True Negative Rate	1.2 minimum duration, 90% intensity threshold	1.2 minimum duration, 90% intensity threshold
Classifier with Lowest True Positive Rate	1.2 minimum duration, 95% intensity threshold	1.0 minimum duration, 95% intensity threshold
Classifier with Lowest True Negative Rate	1.0 minimum duration, 95% intensity threshold	1.0 minimum duration, 95% intensity threshold

4 Conclusion and Future Work

In summary, we described a system which can be successfully used to reduce stuttered speech disfluency by removing certain potentially undesired speech segments. The dataset consisted of speech and speaker metadata and speech signal statistics. The remedied speeches showed a marked improvement over the original speeches.

The innovation in our system is two-fold. First, both the scope of data collection and the classification is predetermined and designed with the singular objective of determining whether a possible action must be performed or not; in our case, we only collected potentially undesired candidate segments that might be removed, and extracted features which are then used to classify the segments with the sole purpose of deciding whether a segment needs to be removed or not. Secondly, the data collected represents different examination perspectives of single instances.

As a practical enhancement to our system we implemented a voting based classification system to reduce the possibility of speech meaning loss by tilting the model in favor of less meaning loss at the expense of missed removal of blocked segments. This also was interesting in outcome and continued such explorations would likely bring practical usability enhancements to the system.

A specific use of a system such as the one presented in this work is to remedy a disfluent speech with blocks for a speaker who wishes to make his (or her) speech easy to listen to and better understood. Remediation makes a speech easier to listen to with minimal inconvenience to the listener and minimal need for re-recording.

The concept of single action based intelligent solutions can help systems utilize compartmentalized machine learning and classification solutions. Such scope-specific machine-learned actions can provide lightweight and fast independent action prediction tool that can be chained together for more complex tasks. The process of utilizing the use of multiple perspectives can be employed to optimize and devise an adaptive approach to data gathering, where the data collection method and scope are optimized according to the metadata and condition of the subject on hand.

In future work, we recommend utilizing various extraction thresholds to fine tune the feature collection to the speech condition, which would better streamline the remediation process. Another extension to disfluent speech remediation would be to eliminate speech interjections by identifying the voiced and unvoiced segments of speech and then singling out those voiced sound segments with interjection episode characteristics for removal.

References

1. Ai, O. C., Hariharan, M., Yaacob, S., Chee, L. S.: Classification of speech dysfluencies with MFCC and LPCC features. In: Expert Systems with Applications, 39(2), 2157-2165 (2012).
2. Boersma, P. "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound." Proceedings of the institute of phonetic sciences. Vol. 17. No. 1193. 1993.
3. Chee, L. S., Ai, O. C., Yaacob, S.: Overview of automatic stuttering recognition system. In: Proc. International Conference on Man-Machine Systems, no. October, Batu F, Penang Malaysia, 1-6 (2009).
4. Fook, C. Y., Muthusamy, H., Chee, L. S., Yaacob, S. B., Adom, A. H. B.: Comparison of speech parameterization techniques for the classification of speech disfluencies. In: Turkish Journal of Electrical Engineering & Computer Sciences, 21, no. Sup. 1 (2013).

5. Hariharan, M., Chee, L. S., Ai, O. C., Yaacob, S.: Classification of speech dysfluencies using LPC based parameterization techniques. In: Journal of medical systems, 36(3), 1821-1830 (2012).
6. Honal, M., Schultz, T.: Automatic Disfluency Removal on Recognized Spontaneous Speech-Rapid Adaptation to Speaker Dependent Disfluencies. In: ICASSP (1), 969-972 (2005).
7. Honal, M., Schultz, T.: Correction of disfluencies in spontaneous speech using a noisy-channel approach. In: Interspeech (2003).
8. Howell, P., Davis, S., Bartrip, J.: The UCLASS archive of stuttered speech. In: Journal of Speech, Language, and Hearing Research, 52(2), 556-569 (2009).
9. KM, R. K., Ganesan, S.: Comparison of multidimensional MFCC feature vectors for objective assessment of stuttered disfluencies. In: International Journal of Advanced Networking Applications, 2(05), 854-860 (2011).
10. Lease, M., Johnson, M., Charniak, E.: Recognizing disfluencies in conversational speech. In: IEEE Transactions on Audio, Speech, and Language Processing, 14(5), 1566-1573 (2006).
11. Liu, Y., Shriberg, E., Stolcke, A., Harper, M. P.: Comparing HMM, maximum entropy, and conditional random fields for disfluency detection. In: Interspeech, 3313-3316 (2005).
12. Ravikumar, K. M., Rajagopal, R., Nagaraj, H. C.: An Approach for Objective Assessment of Stuttered Speech Using MFCC. In: The International Congress for global Science and Technology, p. 19 (2009).
13. Świetlicka, I., Kuniszyk-Jóźkowiak, W., Smołka, E.: Hierarchical ANN system for stuttering identification. In: Computer Speech & Language, 27(1), 228-242 (2013).
14. Raghavendra, M., P Rajeswari: Determination Of Disfluencies Associated In Stuttered Speech Using MFCC Feature Extraction. In: Computer Speech & Language, IJEDR, 4(2), 2321-9939 (2016)
15. Czyzewski, A., Kaczmarek, A., Kostek, B. Intelligent Processing of Stuttered Speech. J. Intell. IN: Inf. Syst.. 21. 143-171 (2003)