Detection of stuttering problems

Eynav Ben Shlomo[1], *Elona Mendes-Ossona*[2]

[1]School of Computer Science, Ariel University, Israel

eynavbe@gmail.com[1], elonamin@gmail.com[2]

Abstract

The article presents a way of classifying stuttering in speech. The method is a way that combines audio and text clips and uses acoustic and linguistic characteristics related to stuttering.

The method is performed by using the Wav2vec model to convert audio to a matrix representation, and Agnostic BERT to create vector representations from the transcripts. These represent sentences are concatenated to form a combined vector. A binary classifier then uses the data to distinguish between "stutterers" and "non-stutterers". If he stuttered, the percentage of stuttering, the signs of stuttering were obtained and the seconds during the audio in which stuttering was detected.

[1]

1. Introduction

Pronunciation problems and stuttering is a common issue in connection with many researchers in various fields. We try to use the analysis of audio clips to check whether a person has a pronunciation defect or a stutter and with the help of our analysis we give feedback to the speaker so that he can learn from it and improve his speech.

Before delving into the details of the work in the article, below is an explanation of the phenomenon that the article discusses. [2]

Pronunciation is the production of sounds of the language. Mispronunciation is producing the vowels of the letters inaccurately. There are several levels of pronunciation, level 1 pronunciation is incorrect pronunciation, the speaker is not understood with an emphasis on grammar and vocabulary. Pronunciation at level 2 The speaker is understandable but the pronunciation is unacceptable because he has a strange and heavy accent. Pronunciation at level 3 Good pronunciation, in the usual way to pronounce the words.[1]

There are various types of speech disorders, including dysarthria, apraxia, cluttering, lisping, and stuttering. Dysarthria is a speech disorder caused by muscle weakness, while apraxia is a speech disorder that occurs when the neural path between the nervous system and the muscles responsible for speech production is obscured or lost. Cluttering is characterized by speech that is too jerky or rapid, while lisping is the incapability of producing sibilant consonants correctly. Stuttering is a neuro-developmental speech disorder characterized by involuntary stoppages, repetition, and prolongation of sounds, syllables, words, or phrases. It is the most common speech disorder and can be classified as developmental or neurogenic. Developmental stuttering occurs during the learning phase of the language, while neurogenic stuttering occurs after head trauma, brain stroke, or any kind of brain injury.

Stuttering is a disturbance in the rhythm of speech that occurs when the person knows what he wants to say but the sentence he speaks comes out with an involuntary prolongation, repetition or pause of a sound.

According to The Stuttering Foundation More than 80 million people worldwide stutter, which is about 1% of the population.

There are 2 main types of stuttering, developmental or acquired. Developmental stuttering is caused by language learning in childhood. Acquired stuttering is caused by psychogenic reasons due to emotional or physical trauma. Acquired stuttering can also be caused after a neurological injury. In this article we will focus on stuttering caused by neurological damage. Neurogenic stuttering is of low frequency, when it is more common in men between 2:1 and 10:1. neurogenic stuttering has been reported in patients with stroke, traumatic brain injury, epilepsy, multiple sclerosis, Parkinson, corticobasal ganglionic degeneration, senile dementia, dialysis dementia, hypoxic ischemic encephalopathy, and pharmacological iatrogenesis.

Stuttering is a speech disorder characterized by repetitions, prolongations, and blocks of sounds, syllables, or words. It is caused by the disruption of the normal speech production process, which affects the fluency and rhythm of speech. Research shows that during stuttering, there is a dramatic change in cerebral activity. The right hemisphere areas become active, and the left hemisphere areas become less active. PWS also tend to develop strategies allowing them to avoid sounds or words which can result in disfluency; such strategies consist in using paraphrases or synonyms instead of the problematic segment. [2]

The negative effects that can cause a speaker with stuttering and/or incorrect pronunciation that may result in a negative judgment regarding his personality and ability, It can harm the speaker academically, relationships, socially and professionally.[3]

There are different methods for treating stuttering, medicinal methods that we will not elaborate on, and non-medicinal methods, one of the techniques is speech therapy. [4] In this article, we will expand on this treatment method and offer a treatment for stuttering and incorrect proofreading that uses this method.

There are cases where after a neurological injury there is facial paralysis or facial movement dysfunction, some of the stuttering cases discussed here are stuttering that happens at a later stage of rehabilitation from the injury.

Another study we are conducting discusses the rehabilitation in the first stages after neurological injuries by practicing the facial expressions of the patient who trains Neuromuscular in the face and the patient will receive feedback to advance the treatment.

Analysis of stuttered speech includes (a) mean duration of sound/syllable repetition and sound prolongation, (b) mean number of repeated units per instance of sound/syllable and whole-word repetition, and (c) various related measures of the frequency of all between- and within-word speech dysfluencies. The intensity and time of speech can prove the presence of stuttering. Stuttering disorder is characterized by disruptions in the

production of speech sounds, called disfluencies. [5]

Automatic speech recognition systems (ASR) struggle to recognize stuttered speech, making it difficult for people who stutter (PWS) to use virtual assistant tools. Current speech recognition systems have good accuracy for fluent speech but fail to detect stuttered speech, i.e. if there are repetitions or long pauses in speech. This is because the systems have been programmed to stop process detection when a pause is encountered and in addition these systems are trained on non-stuttered speech.

This highlights the need for automatic stuttering identification systems (ASIS).There are systems that have tried to analyze stuttered speech with technologies such as Artificial Neural Network (ANN), Hidden Markov Model (HMM), Support Vector Machine (SVM) and advanced Other Digital Signal Processing techniques but the results are not good. [6]

The way will assess stuttering and mispronunciation in speech will be by transcribing the recorded speech and analyzing it by repeated occurrence of vowels, vowel prolongation, and improper articulation of movement.

recognition systems using different models, databases, feature extraction methods and classification techniques.

Fig. 1 and Fig. 2 are an example of a text spoken by a non-stuttering speaker in 1 and said again by a stuttering speaker in 2.

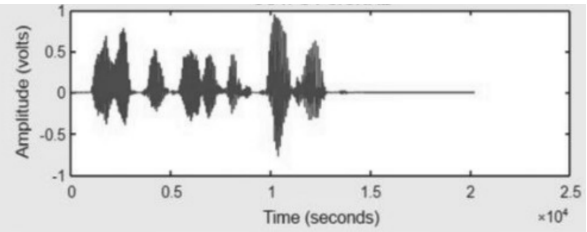

Figure 1: *An example of non-stuttering speech.*

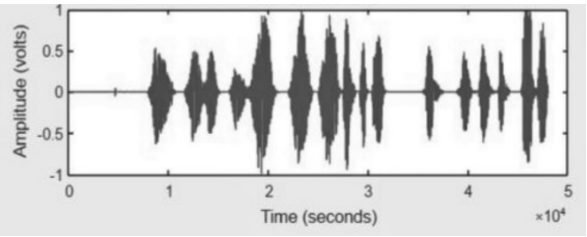

Figure 2: *An example of stuttering speech.*

[7]

We will use a model for learning cross-linguistic speech representations by training on speech from the language,Using voice clips to assess stuttering versus normal speech. The model is based on wav2vec 2.0. Wav2vec is a deep learning model for speech recognition, the model can, with the help of unsupervised learning techniques, recognize spoken words and transcribe them.

the speech therapy sessions are private and are very expensive, thus making it unaffordable to some. [1]

1.1. Our contribution

This paper propose a different approach that combines audio and text representations for the classification of stuttering instances and to help people with stuttering to get feedback on their inaccuracy, practice their speech and improve it. Our algorithm classifies speech into two categories: stuttering and non-stuttering and knows how to identify where the stuttering is said. Training and testing the classifier is done with two different databases, the first one with audio clips of stuttered speech and the second with audio clips of people with pronunciation problems. Since the type of database used is not common, the total number of audio clips is about 1000 audio clips of no more than 5 seconds each, in English. The model achieved 68% accuracy in identifying stuttering and pronunciation problems in the English language. We assume that the accuracy of the algorithm will improve once more suitable sound clips are available. Overall, the proposed work contributes to the field of stuttering detection and gives the person who stutters an opportunity to improve their speech by pointing to the location of the stutter. [1] [2]

1.2. Paper Structure

The remainder of this paper is structured as follows: Section 2 reviews related works mainly regarding existing methods for detecting stuttering and pronunciation problems and previously proposed algorithms that offer a solution; Section 3 discusses the datasets used in this work; Section 4 presents the framework of the proposed method, the tools we used in our approach, including Wav2Vec2 and Bert architectures; Section 5 presents the methods we used to arrive at a solution, detailing each step and how it is used and applied by the user; Section 6 presents the experiments that were done until reaching the final solution, after that we will present the evaluation process of our results and a summary of the final results; Finally, Section 7 concludes the paper and suggests some directions for future work and possible improvements. [2]

2. Related Work

Below are various works dealing with stuttering and pronunciation problems, works and studies done in the past reinforce the need to identify stuttering and pronunciation problems and in a way to monitor changes and improvements in speech. In [1] and [3] a background on stuttering and pronunciation problems is presented. [1] presents explanations about correct pronunciation, how to teach it correctly and its importance, while in [3] we discuss the problem of stuttering and its possible negative consequences on various areas of life. Both articles show us the possible problem and difficulty when speech is not correct, which shows the importance of identifying correct pronunciation in order to improve it in the future. To investigate the phenomenon of stuttering we based ourselves on the research done in [2]. In [2] a detailed explanation of stuttering and different types of stuttering, analysis of stuttering according to the voice segments and treatment of neurological stuttering is presented. The article reinforces that the sound of stuttering is described in the various voice segments that we rely on in the article to check between stuttering and fluent speech, in addition it is stated that the treatment of stuttering can be done in a non-pharmacological way. We aim to improve stuttering in people suffering from both neurological and non-neurological stuttering, by stimulating the muscles of the facial nerve and training in speech, the stimulation will be done by repeating over and over words or sentences that our model will see as ”not good” and thus practice. The detection of stuttering in different voice segments will be done by identifying stop gaps, syllable repetitions and vowel extensions which are signs of stuttering [4].

Other attempts have been made in the past to perform recognition of stuttered speech using a model instead of manually by a person. Previously built models are Surya and Varghese [5], the model converts the audio clip into an audio array and obtains the sound frequency from them, which is used to train the machine. In the presented model, SVM classification is performed that classifies the stuttered input into a correct word by binary representation. The accuracy obtained is 76% in correct classification of the words, the accuracy obtained is relatively high, but the problem with the model is that only the trained words can be predicted. Another approach to stuttering speech detection included the use of a Linear Predictive Cepstral Coefficient (LPCC) algorithm with k-nearest neighbors (k-NN) and linear discriminant classifiers (LDA), which yielded an accuracy of 88.05%. But, these approaches were developed only to identify and classify the known stutterers and do not deal with housing problems in general that also include poor pronunciation.

A method using an algorithm using neural networks along with some proposed string operations has also been developed to recognize the speech of a person who stutters [7]. This algorithm detects the speech of someone who stutters and does not classify whether the person speaking stutters or not, but it is information that is known in advance.

In [6] stuttering detection and classification methods using machine learning and deep learning were reviewed as ways to detect stuttering.

Other proposed works on the subject are citearticleStutter, [8]. In [9] the authors dealt with the identification of stuttering and the classification of stuttering into its subcategories. To this end, the authors tested algorithms and several deep models, including Wav2Vec2, on a stuttering dataset. The article showed that deep learning is superior to the classical approach to stuttering detection, such as detection by speech therapists for example. In [8] the authors present a new stuttering detection system called StutterNet. The system is based on deep learning capable of detecting different types of interference, the proposed method relies only on the acoustic signal, unlike other works that used automatic speech recognition (ASR). They used a time-delay neural network (TDNN) and a deep neural network (DNN) to detect stuttering directly from speech. As in the work presented here, the authors aim to give feedback to people who stutter about their fluency. Although the authors of the paper agree that existing language recognition models are effective for stuttering detection and achieve good results, there is an ambition to find another way because relying on ASR makes the model cost expensive and prone to errors. In the proposed work, unlike [8], we aim to use ASR so that we have fewer errors, in a more efficient way.

ASR has already been used in the past in combination with another factor in order to identify and classify stuttering into categories [10]. The other factor used were comments from the stuttering diagnoses. In [10], the Speech Language Pathologists (SLP's) take notes in real time, while a person speaks and reads a story that is known to be used to diagnose stuttering, and use 8 categories to classify stuttering. The story being read is known in advance. Using ASR to transcribe the words ends up merging the SLP notes with the transcription and getting that there are words to speak versus the words as they were spoken. By comparing, you see what needs correction and correct the transcription that is being made and it is not good. In contrast to them, we use two models without the intervention of a human factor in the stuttering detection. In addition, while training and testing the model, although we know the spoken text, we do not use it for classification purposes, and during the operation of the model it simply flows, the text is unknown to us, and yet we want to search for the speaker's speech.

Another use made using ASR is the detection of stuttering in children [11]. Since ASR outputs a clean and legible output, it is not always possible to detect when one is speaking in a stammering manner, and therefore an ASR system is needed that outputs the speech in a manner as similar as possible to the speech of the sources, without corrections. Indeed, after "fixing" the ASR system, the authors were able to identify more cases of stuttering.

The last two works mentioned above emphasize the efficiency and advantages of using the ASR system for stuttering detection.

In the proposed work wav2vec [12] and Agnostic Bert are used. [13] discusses wav2vec 2.0 in cross-linguistic learning in speech processing and focuses on unsupervised representation learning, which does not require labeled data. Learn representations on unlabeled data that generalize across languages. [12], [13] were used to understand the possibilities of using Wav2vec for the current work. In our work we use sentence embeddings using BERT which is language agnostic [14]. In addition, we used the loss function which is a simple and effective auxiliary function to improve speech recognition using the CTC framework [15]. In order to measure the performance of machine learning, accuracy, F1-score, precision and recall were used. Each has its advantages and disadvantages in performance measurement, and they are different from each other [16][17].

In our work, we will see the detection of stuttering and pronunciation problems using deep learning that classifies vectors provided by Wav2vec and Bert of audio segments into stuttering/non-stuttering. [1] [2]

3. Datasets

Datasets are necessary to train the machine learning model. The dataset we use in this work is based on the apple's ml-stuttering-events-dataset and the TORGO [18] dataset. All audio files in the datasets are no longer than 5 seconds and the number of audio samples that are taken per second is 16,000 Hz.

3.1. TORGO

TORGO is a free database intended for research purposes. TORGO was developed by a collaboration between departments of Speech Language Pathology Computer Science at the University of Toronto and the Holland-Bloorview Kids Rehab hospital. The TORGO database was originally intended to be a resource for developing automatic speech recognition (ASR) models to improve voice recognition for people with dysarthria. People with dysarthria often have atypical and less intelligible speech than normal speech, so traditional ASR software has a harder time working optimally with people with this problem.

Various sound clips were collected for the database according to the pre-prepared texts, which are also included in the database. The text segments had non-words, short words, restricted sentences and unrestricted sentences in order to identify a difference between speakers who have dysarthria and those who do not have dysarthria in different sentences and contexts.

The data was collected according to the gender of the speaker, whether he suffers from dysarthria and the number of the session in which the speaker recorded himself saying the given texts. This dataset comprises samples from seven persons, diagnosed with cerebralpalsy or amyotrophic lateral scle-

rosis including four males and three females aged between 16 to 50 years. In addition to this, it also contains samples from control speakers of the same age.

Table 1: *Summary of the amount of data in the TORGO dataset*

	Female	**Male**	**Total**
Dysarthria	1430	1623	3053
Without dysarthria	2512	3724	6236

[2]

3.2. SEP-28k

The apple dataset we used,SEP-28k, is free to use, modify and distribute under certain conditions listed therein. The dataset based on the podcasts retains the copyright of the podcast owners. The SEP-28k contains about 385 podcasts of various lengths in English by people who stutter and have pronunciation problems. The podcasts are from a variety of people of different genders with different voices and speaking styles for classifying people with stuttering and/or pronunciation problems. There are links to podcasts from which you can reach the desired audio file. In order to train the model, audio clips of people who stutter and audio clips of people who do not stutter were needed, but in SEP-28k there are only audio clips of people who stutter, so it was necessary to create a similar dataset of people who do not stutter.

The steps of building the dataset of non-stuttering people: 1. First we divided selected podcasts into equal segments of 5 seconds each to more accurately identify the place of the stutter in the audio segment.

2. The audio segments were tested using wav2vec [12]. Each audio segment was sent to wav2vec which transcribed the segment, then a comparison was made with the output (to the transcription of the audio segment) and thus it was determined whether the current audio segment is of a person who stutters. If it was found that the person in the recording stutters, the recording and its transcription were saved.

3. Creation of a new data set: based on the saved recordings and transcripts, each piece of text belonging to the recording was transferred to ttsmp3. ttsmp3 is a free site that allows you to convert text to mp3 audio files in a variety of languages. In order to create audio segments of recordings of normal speech, we used the conversion of text into the English language in two different genders, male and female, and thus created a new data set containing audio files of the same text with normal speech. The dataset is divided equally into 50 percent male and 50 percent female. [2]

4. Framework

Several different tools and models were used for this work, among them Wav2vec, Bert, CTC loss and more, which we will detail now.

4.1. wav2vec 2.0

Wav2Vec is an approach developed by Facebook AI Research. Wav2Vec, unlike traditional automatic speech recognition (ASR) approaches that use large amounts of transcribed audio data for training, is a model that pre-trains a model on a large unlabeled dataset consisting of raw audio waveform data. Based on the past audio, the model knows how to predict the future waveform samples. Thanks to this, Wav2Vec can learn representations of speech without relying on transcriptions. After pre-training, the model is fine-tuned on a small transcribed data set with labels, so the model can adapt to specific tasks.

The Wav2Vec2 model was trained using connectivist temporal classification (CTC) and therefore the model output must be decoded using the Wav2Vec2CTCTokenizer which we will detail in 6.1.1.

Wav2Vec has achieved advanced performance in various measures of speech reconization and eliminates the need for feature extraction.

Here is a figure describing the wav2vec process framework:

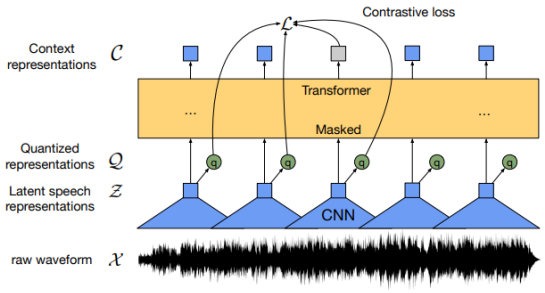

Figure 3: *The stages of the speech representation in wav2vec. The framework jointly learns speech representations in contexts of speech units.*

We now explain Fig.3: Raw waveform, represented by X, is the waveform that represents the original audio signal before processing. CNN us a deep learning architecture for audio transition and waveform processing. Latent speech representations, represented by Z, is a loss function that is used during training so that similar speech segments are close to each other. Quantized representations, represented by Q, are the representations of the important features in the speech learned by the model. Masked is when a target is missing and it is used to fill in the gap. Transformer is a type of neural network architecture to understand the different parts of speech. Context representations, represented by C, is the representation of the speech according to the context when according to the context of the speech it can improve the understanding of the speech, by this information is obtained about the words before and after the speech segment. Contrastive loss is the neural network architecture used for learning speech that tests the dependencies between parts of the input. [12]

4.1.1. Wav2Vec2Tokenizer

Wav2Vec2Tokenizer is a tool utilized in the field of audio data processing. Its primary purpose is to preprocess audio data, including speech or sound, before feeding it into the Wav2Vec2 model. This tokenizer plays a crucial role in tasks such as speech recognition or speech classification by carrying out essential functions like tokenization, encoding, and handling of audio input length. By utilizing the Wav2Vec2Tokenizer, audio data can be appropriately prepared and transformed into numerical representations suitable for analysis and processing by the Wav2Vec2 model.

4.1.2. *Wav2Vec2ForCTC*

The Wav2Vec2ForCTC is a model architecture for speech recognition tasks using the Connectionist Temporal Classification (CTC) loss function. The model takes received audio segments as input and produces transcriptions or labels for the received speech. It is trained in a self-supervised manner using large amounts of unlabeled audio data and uses a CTC-based objective function to learn to align the predicted labels with the audio input. [12][13] [2]

4.2. Agnostic BERT

In our work sentence embedding using language-agnostic BERT [14]. Bert was developed by Google AI Language in 2018. Bert is short for Bidirectional Encoder Representations from Transformers, it is a machine learning model for natural language processing. Bert is a revolutionary innovation in the field of NLP because it improved results in more than 11 common tasks in the field of natural language processing. With the help of Bert, we can analyze sentiments, answer questions (with the help of bots), predict text, create text, summarize sentences and texts, classify words into categories according to the context, and more. BERT can adapt to multiple languages without the need for language-specific training.

BERT has over 3.3 billion words that contributed to its training data, in addition it was trained on large datasets, such as Wikipedia for example, which contributed to its deep learning. BERT training was made possible with the Transformer architecture. The Transformer architecture allows to perform machine learning (ML) training in parallel in an efficient manner, therefore the reception of training allows BERT to be trained on large amounts of data in a relatively short time. BERT's source code is publicly available.

Agnostic-BERT is an approach whose goal is to make a model more neutral by reducing the biases of certain features or characteristics in language models such as BERT, feature bias can arise from unbalanced training data. To create an agnostic bert model there are different methods, among them a careful selection of data.[14] [1] [2]

4.3. CTC loss

CTC learns the alignments between speech frames and label sequences. CTC was proposed for encoding sequence labeling with inputs and outputs of variable length, with empty symbols sandwiched between the labels. CTC has not been fully researched on languages other than English. The CTC classifier is a loss function for sequence labeling problems where the input and character sequences have different lengths.

CTC loss calculates the explanations for an output sequence given an input sequence. CTC introduces an intermediate representation called a CTC path, this path is a sequence of characters at the frame level, which allows the occurrence of blank characters and repetitions of the non-blank characters, the blank character marks the absence of an element or a gap between consecutive elements. [15]
[1]

4.4. Feedforward neural network with fully connected dense layers

In this work, the Feedforward neural network with fully connected (dense) layers model was used to classify the data. Artificial neural networks are a type of machine learning inspired by the neurons of the human brain. In the last decade, there has been an increase in the use of neural networks due to higher availability of data and higher computing capabilities, because to build a deep learning model based on neural networks requires a lot of information and computing capabilities. There are several types of neural networks, for example RNN and feedforward neural networks (FNN) but we will focus on FNN which is used in our model. FNN has been applied to pattern classification, regression problems, optimization, control and prediction, and more. The neurons in the human brain, which served as inspiration for FNN, are structured so that each neuron has a cell, and has two branches so that one receives signals from the previous neurons and the other transmits the signals to the following neurons. The neurons in the brain also have a synapse between two neurons, which can enhance or inhibit signals that pass through it. If the signals of a certain neuron cross a certain threshold, it transmits the signal, otherwise the signal will not be transmitted.

The oldest neural network in use today is the Perceptron. This network is a form of feedforward neural network. It receives several inputs, each cell (neuron) has a weight that represents the synapse, so that each incoming input is multiplied by the weight of the cell and the inputs arriving at the neuron are combined into one input. As happens in the brain, a certain threshold is needed for the neuron to transmit the input, this is the activation function. For each input obtained from the multiplication of the cell weight and the connection between the inputs, a bias value is added to the "final" result and only then the final value of the cell is passed to the activation function, the bias allows the activation function to be changed. The activation function works so that if the final result is greater than zero, the value of the neuron is passed on, otherwise if the value is 0, the value of the neuron is not passed on.

For the learning of the machine, pairs of input and output are given and thus the algorithm knows how to adapt itself to the input so that the output that comes out will be the same as the given output. In the model used in the article, the Softmax activation function is used, this function takes all the neurons in the output layer and creates a probability distribution so that the sum of the values of all the output neurons will add up to 1. The neuron with the highest value, whose predicted probability is the highest, will indicate the predicted label .

Hidden layers in FNN are the layers found between the input layer and the output layer, they allow to solve problems where the data cannot be separated in a linear way.

The number of neurons in the output layer depends on the type of problem the neural network wants to solve. [19]

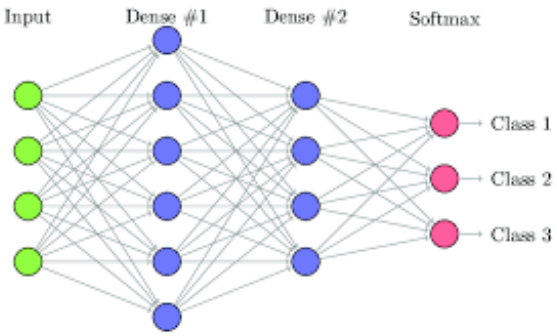

Figure 4: *Example of fully-connected neural network.*[1]

[2]

5. Algorithm flow

Description of how the algorithm works according to the stages of execution from the moment the data is loaded to the final result:

5.1. Importing the data

Importing the pre-prepared data that is classified as stuttering or non-stuttering. According to the path of the root folder of the stutter folder to the non-stutter folder, all wav files in the root folders are imported. And the list of the wav files found matching stuttering and non-stuttering audio files is returned. Each of the audio files from the found path goes through several steps. [1]

5.2. Transcription: Transcription of the audio file

Transcript of the audio file obtained using the Wav2Vec2 model and tokenizer. The pre-trained model "facebook/wav2vec2-base-960h" was used. When getting the audio file path, torchaudio is used to load the audio file and get the waveform over time of the audio and sample rate. If the sampling rate of the audio is not 16000Hz, the audio format is converted to the sampling rate of 16000Hz, the number of samples per second, in the audio file the waveform is sent to a tokenizer that breaks the audio waveform into smaller blocks or tokens based on its acoustic software and returns values to be sent to the Wav2Vec2 model that is trained to predict the sequence The speech units from the input audio, Wav2Vec2 which creates probabilities representing the predictions for each possible token passing each position in the audio in the input, to find the most likely token in each position in the input audio sequence, a search is performed to find the index with the highest value along the index in the matrix. And an array of predicted token identifiers is obtained where each identifier represents the index of the token with the highest probability for each position in the input. The predicted transcript for the input audio is sent to a decoding tokenizer, which converts the token identifiers into readable text by matching the tokens to their corresponding vocabulary. and the text of the audio file transcript is obtained. The transcription is with the errors caused by the stuttering such as repeating characters or expressing a character improperly. [1]

5.3. Wav2Vec: Sound to Matrix Representation

The project uses the Wav2Vec model. Wav2Vec takes audio waveforms as input and produces a sequence of feature vectors. By converting the sounds into a matrix representation, thus capturing relevant vital information of stuttering.

wav2vec is a model that is a network that takes audio as input and calculates a representation that can be fed to a speech recognition and understanding system. wav2vec is trained by large amounts of audio data.

When speech is made or a certain sound is heard, a pattern is created in the form of a wave that represents the sound in the audio. Wav2Vec2 uses the form of the wave where each segment of the wave represents numbers that represent the audio character, Wav2Vec2 uses a deep learning architecture known as Convolutional Neural Network (CNN) to process the audio waveforms. The waveform is divided into a short segment of the audio note, for each of them Wav2Vec2 calculates the different characteristics of the wave. The calculated features are transferred to a CNN which is a neural network that can learn patterns from input data and it identifies the certain structures present in the audio and it goes through another stage where the important information is selected and at the end the features are coded into a matrix representation. The matrix represents the important features and characteristics of the audio signal. Wav2Vec2 with which you can analyze and understand the audio signals in a more efficient way.

Used to convert audio files to matrix representations, this matrix representation facilitates the extraction of features responsible for processing the waveform from the audio, from which important features about the audio can be learned. The sampling rate at which the audio will work is 1600Hz. The path of the audio file received as input is loaded and the audio waveform and sample rate data is returned. If the sampling end is not 1600Hz, conversion is performed by resampling the audio and producing a version with the sampling rate of 1600Hz. The importance of the 1600Hz sample rate before extracting the audio features is because audio processing models require this sample rate to work effectively. Extracting the audio features using a Wav2Vec2 feature extractor, getting the path of an audio file and applying an extractor component defining the feature extraction from the audio, using the Wav2Vec2 model. The matrix represents the audio in a compressed form suitable for further analysis.

[1]

5.4. Matrix for one vector

Converts a matrix to a one-dimensional vector by calculating the average values along the columns. For each column in the matrix, the average value is calculated and a vector is obtained that represents the average values in the matrix. [1]

5.5. Agnostic BERT: Transcript to vector embedding

Agnostic BERT is used which creates vector embedding vector embedding transcribers represent a vector representation of the input text and encode syntactic information. Using this to capture the linguistic features of the input text.

Generates a vector from the transcript using Model Agnostic BERT. A transcript is received as an input that is the representation of the audio file, the transcript contains the errors caused by the stuttering, and returns the vector from the transcript. Agnostic BERT A model trained on a large text that allows receiving all kinds of data according to the text. The text undergoes a splitting process into tokens or individual words and apply the pre-processing steps using a tokenizer. The token text is passed to the Agnostic BERT model and returns the outputs by the Agnostic BERT model. The outputs are a variety of information that are representations learned from the input. and to extract the vector representation of the CLS token representing the total representation of the transcript from the outputs of the model. The CLS token is used in the Agnostic BERT model when a single vector summarizing all the information of the transcript needs to be presented. [1]

5.6. Vector Concatenation

The matrix representation from Wav2Vec and thevector embedding from Agnostic BERT are concatenated to create a com-

[1] Source:Temporal Convolutional Neural Network for the Classification of Satellite Image Time Series - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Example-of-fully-connected-neural-network$_{f}ig2_{3}31525817[accessed12Jun, 2023]$.

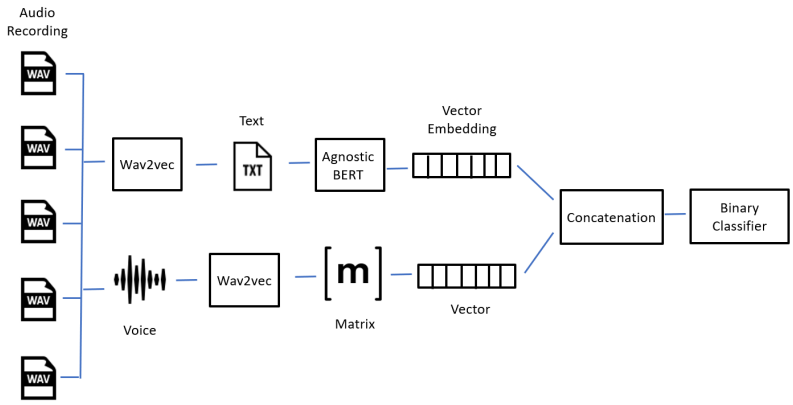

Figure 5: *The steps through which the audio passes through the model for training.*

bined vector of audio and text. This approach allows linking both acoustic and linguistic information and will improve classification performance.

Concatenating two vectors and creating one combined vector that contains information from both vectors. One vector from the vector created after merging the Matrix from Wav2Vec into one vector, the second vector is the vector created from Agnostic BER. Accepts as input the two vectors. It is done by getting the length of each of the vectors, a check is made that the vector received from Wav2Vec is indeed one vector and therefore corresponds to the vector created from Agnostic BER. Concatenate the vector created from Agnostic BER that will repeat itself twice according to the length of the vector received from Wav2Vec and the vector created from Wav2Vec as a result a new vector is created. [1]

5.7. Binary Classifier Training

After each of the audio segments of the data goes through each of the stages, the result is sent to classifier training, high-level neural networks.

From the collected databse of audio segments marked with stuttering and audio segments marked as without stuttering. Using a feature vector the combined audio and text will be as input to the classifier. Train the binary masker to distinguish between a stutterer and a stutterer. We will use a logistic regression machine learning algorithm in train.

The input is a list of the vectors obtained from the steps and a list of their classification 0 or 1 which shows as non-stuttering or stuttering respectively. Building the deep learning training neural network model by the Sequential model with three dense layers relu and an output layer sigmoid , the first dense layer includes 64 units and is set to learn and capture non-linear relationships between the input vectors and the target labels, adapted to the input vector metrics. The second dense layer includes 64 units to improve model capability and avoid overfitting by encouraging smaller weight values. Using Adam is used for binary classification, and follows the accuracy index. The model is trained for 20 epochs with a batch size of 32. Then saves the trained model to a file, so that later the saved model can be loaded to make predictions on receiving new audio if stuttering is detected in the audio by the classifier.

A part of the dataset is sent to train the classifier.
[1]

Table 2: *Summary of the amount of data in the train dataset*

	stutter	**No stutter**	**Total**
audio files	40	40	80
seconds	200	80	280

5.8. Sending to classification

The input is the concatenated vector in running the previous steps, running it on the saved classifier obtained in running Binary Classifier Training and getting the classification of 0 or 1 that indicates stuttering or non-stuttering. [1]

5.9. Stuttering seconds

Identifying the seconds that the stutter occurs in an audio segment using the Wav2Vec model and CTC loss. Accepts as input the path to the audio file and the target text that the speaker intended to say. Using the pre-trained Wav2Vec model, load the audio file, the audio file should have a sampling rate of 16000Hz, if it is not then the file is resampled to match the sampling rate. The audio file is transferred to receive the waveform and receive the input values using Wav2Vec. To obtain the loss CTC, the probability of each token is obtained at each stage during the audio, the ID of the target transcript is adjusted according to the specific token in the vocabulary. loss CTC detects the level of mismatch between the speech audio and the target speech audio.

To detect possible stuttering in the audio and the seconds in the audio that it happens, then the algorithm uses non-blank tokens to detect stuttering segments by having the code check if a token is not equal to the blank token identifier and not equal to the previous token, by tracking non-blank tokens that indicate the beginning of speech and tracking The duration of non-empty token sequences and if the duration between start and end exceeds the threshold then the code identifies this as possible stuttering segments. The size of the CTC loss and the time intervals where stuttering was detected in an audio file are returned, the time intervals are represented as start and end seconds. [1]

5.10. Audio-detected stuttering segments

To identify the duration of the seconds in the video where stuttering is detected, loss CTC is used. The CTC loss measures the dissimilarity between the predicted sequence and the target sequence . It takes into account the possible alignments between the two sequences and allows for the presence of blank tokens, which represent unaligned positions. changes in the predicted token IDs after encountering a blank token. The changes could indicate the start or end of a stuttering interval. If it exceeds the threshold, then stuttering is detected.

5.11. Stuttering detection from transcription

Provides information about the stuttering in an audio segment by comparing the transcript from the audio with the expected transcript. Identifies areas where the transcription from the audio differs from the expected transcription indicating possible stuttering. Wav2Vec2 is used for audio transcription,

Identifying the sections where stuttering occurs in the audio section, by transcribing the audio and comparing it to the expected transcript,

two transcribed segments are sent to the LCS algorithm To find the maximum common characters between two segments A comparison is made between transcribing the audio and the segment obtained from the LCS algorithm,the different characters indicate possible stuttering. Any non-matching segment is added as a stuttering segment.

Calculating the percentage of stuttering by the characters who stutter in which the sum of the character lengths in the stuttering segments is calculated and divided by the expected transcript length.

Returns the stuttering percentage, the stuttered characters, the transcription from the audio and the expected transcription. [1]

5.12. User side: training and improving speech

A person who experiences speaking with a stutter will be able to practice speaking and improve it with the help of the feedback he will receive. With the help of a platform the user will fill in the details of the target text he wants to say and record himself for 5 seconds or alternatively upload a wav file of the recording with the stuttered speech. By clicking send, the information will be sent for analysis.

The audio will go through the steps listed before the audio is sent to (1) Transcription: Transcription of the audio file and then Agnostic BERT: Transcript to vector embedding And at the same time the audio will also be sent to the stages (2) Wav2Vec: Sound to Matrix Representation and then Matrix for one vector The output of 1 and 2 will go through the Vector Concatenation step and then send to the pre-trained classifier file to get a binary answer of 0 or 1 of not stuttering or stuttering. Upon receiving an answer from the stutterer, the user will receive feedback.

The feedback received is the seconds during the time of the audio where stuttering was found, the percentage of stuttering detected, the signs that the speaker had a stuttering problem. [1]

6. Experimental Evaluation and Results

After training the model and performing the test, we achieved an accuracy of 68 percent. Below is an explanation of the test steps and how to measure performance.

In tests of the effectiveness of the approach in the article. In order to calculate the accuracy results, indices were used accuracy, precision, recall and F1-score.

A part of the dataset is sent to test the classifier.

Table 3: *Summary of the amount of data in the test dataset*

	stutter	**No stutter**	**Total**
audio files	11	11	22
seconds	55	27	82

Each index provides different insights into the model. Accuracy provides a general assessment of the accuracy of the model.

TP: True Positives, correctly predicted positive instances.

TN: True Negatives, correctly predicted negative instances.

FP: False Positives, incorrectly predicted positive instances.

FN: False Negatives, incorrectly predicted negative instances.

$$Accuracy(Acc) = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Precision provides the proportion of well-predicted positive cases out of all positive-predicted cases. This is when the result of a false positive is critical and you need to check the amount of its accuracy.

$$Precision(Pre) = \frac{TN}{TN + FP} \quad (2)$$

Recall provides the proportion of well-predicted positive cases out of all actual positive cases.

$$Precision(Re) = \frac{TP}{TP + FN} \quad (3)$$

F1-score provides a measure that also takes into account precision and recall. It combines them so that we find the balance between correctly identifying positive cases and minimizing the wrong prediction.

$$F1 - score = 2\frac{(precision)(recall)}{precision + recall} \quad (4)$$

[16] [17]

The results of the measurement according to the criteria mentioned above.

Experiments were performed with activation functions and the amount of different layers. So the last layer is 'sigmoid'. This indicates that it is a binary classification problem, and the output of the model will be a probability value between 0 and 1. Experiment 1 is 3 layers Softmax and layer 4 is sigmoid the results are: Precision: 0.63 Recall: 0.45 F1-score: 0.53 Accuracy: 0.63 Experiment 2 is 4 layers Softmax and layer 4 is sigmoid the results are: Precision: 0.8 Recall: 0.36 F1-score: 0.5 Accuracy: 0.63 Experiment 3 is 3 layers Log-Softmax and layer 4 is sigmoid the results are: Precision: 0.5 Recall: 1.0 F1-score: 0.67 Accuracy: 0.5 Experiment 4 is 2 layers relu and layer 4 is sigmoid the results are: Precision: 0.80 Recall: 0.36 F1-score: 0.50 Accuracy: 0.63 Experiment 5 is 3 layers relu and layer 4 is sigmoid the results are: Precision: 0.83 Recall: 0.45 F1-score: 0.59 Accuracy: 0.68 Experiment 6 is 4 layers relu and layer 4 is sigmoid the results are: Precision: 0.83 Recall: 0.45 F1-score: 0.59 Accuracy: 0.68

The results of the measurement according to the criteria mentioned above are shown in Table 4.

The results are about 55 seconds of stuttering and 27 seconds of no stuttering. A total of 82 seconds.

[1]

Table 4: *The classifier results*

Accuracy	Precision	Recall	F1-score
0.68	0.83	0.45	0.59

7. Conclusions and Future Work

The results of the study emphasize the effectiveness of the approach in the article for classifying stuttering cases. The approach combined audio and text representations and effectively captured characteristics associated with stuttering. The concatenated vector provided a representation that improves classification performance. But due to the small data on which the classifier was created, the results do not demonstrate the maximum efficiency it could display.

Future work on the subject could be training the classifier on a larger number of data and testing whether the results are better after increasing the database on which the model is trained and tested. In addition, our model is trained for the English language, so a possibility for further future work is to adapt the model to other languages.

[1]

8. References

[1] A. P. Gilakjani, "English pronunciation instruction: A literature review," pp. 1–6, 2016.

[2] G. B. R. N. Cristina Cruz, Hugo Amorim, "Neurogenic stuttering: A review of the literature," 2018.

[3] F. I. A. S. Stephanie Hughes, Rodney Gabel, "University students' perceptions of the life effects of stuttering," *Journal of communication disorders*, pp. 45–60, 2010.

[4] J. VanSwearingen, "Facial rehabilitation: A neuromuscular reeducation, patient-centered approach," pp. 250–259, 2008.

[5] A. K. Andrzej Czyzewski and B. Kostek, "Intelligent processing of stuttered speech," *Journal of communication disorders*, pp. 143–171, 2003.

[6] F. H. S. O. Shakeel A. Sheikh, Md Sahidullah, "Machine learning for stuttering identification: Review, challenges and future directions," *Neurocomputing*, 2022.

[7] T. M. V. Y. Ankit Dash, Nikhil Subramani and S. Tripathi, "Speech recognition and correction of a stuttered speech," *Communications and Informatics*, pp. 1757–1760, 2018.

[8] F. H. S. O. Shakeel A. Sheikh, Md Sahidullah, "Stutternet: Stuttering detection using time delay neural network," *arXiv:2105.05599v2*, Jun. 2021.

[9] F. Piotr and K. Bozena, "Rediscovering automatic detection of stuttering and its subclasses through machine learning—the impact of changing deep model architecture and amount of data in the training set," May 2023.

[10] P. Heeman, R. Lunsford, A. McMillin, and J. Yaruss, "Using clinician annotations to improve automatic speech recognition of stuttered speech," pp. 2651–2655, 09 2016.

[11] S. Al, M. Hasan, A. Simons, S. Brumfitt, and P. Green, "Detecting stuttering events in transcripts of children's speech," 09 2017.

[12] Y. M. A. A. M. Baevski Alexei, Zhou, "wav2vec 2.0: A framework for self-supervised learning of speech representations," vol. 33, pp. 12 449–12 460, 2020.

[13] R. C. A. M. M. A. Alexis Conneau, Alexei Baevski, "Unsupervised cross-lingual representation learning for speech recognition," *arXiv preprint arXiv:2006.13979.*, 2020.

[14] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic bert sentence embedding," *arXiv preprint arXiv:2007.01852*, 2020.

[15] J. Lee and S. Watanabe, "Intermediate loss regularization for ctc-based speech recognition," pp. 6224–6228, 2021.

[16] R. Yacouby and D. Axman, "Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models," pp. 79–91, Nov. 2020. [Online]. Available: https://aclanthology.org/2020.eval4nlp-1.9

[17] T. Set, "Precision recall f1-score auc accuracy," *Network*, vol. 16, p. 12.

[18] F. Rudzicz, "Using articulatory likelihoods in the recognition of dysarthric speech," vol. 54(3), pp. 430–444, March 2012.

[19] S. Hiltemann, H. Rasche, S. Gladman, H.-R. Hotz, D. Larivire, D. Blankenberg, P. D. Jagtap, T. Wollmann, A. Bretaudeau, N. Gou, T. J. Griffin, C. Royaux, Y. L. Bras, S. Mehta, A. Syme, F. Coppens, B. Droesbeke, N. Soranzo, W. Bacon, F. Psomopoulos, C. Gallardo-Alba, J. Davis, M. C. Föll, M. Fahrner, M. A. Doyle, B. Serrano-Solano, A. C. Fouilloux, P. van Heusden, W. Maier, D. Clements, F. Heyl, B. Grüning, and B. B. and, "Galaxy training: A powerful framework for teaching!" *PLoS Comput Biol Computational Biology*, vol. 19, no. 1, p. e1010752, jan 2023.