A logo with text and birds on it

Description automatically generated

Ain Shams University

Faculty of Computer & Information Sciences

Scientific Computing Department

A logo with a check mark and a circle

Description automatically generated

**SpeCor: Speech Recognition and Stutter Correction**

**By**

Abdelrahman Fakhry Hussein

Mohamed Abdelmonem Abdelrahman

Mohammed Mahmoud Hasan

Tarek Abo Elenen Ali

Basma Ahmed Said

Eman Basem Ezzat

**Under Supervision of**

**Dr. Manal Mohsen Tantawi**

Associate Professor,

Scientific Computing Department,

Faculty of Computer and Information Sciences,

Ain Shams University.

**TA. Manar Sultan**,

Scientific Computing Department,

Faculty of Computer and Information Sciences,

Ain Shams University.

**June 2024**

# Acknowledgments

All praise and thanks to Allah, who provided us with the ability to complete this work. We hope to accept this work from us.

We are grateful to our parents and our families who are always providing help and support throughout the whole years of study. We hope we can give that back to them.

We also offer our sincerest gratitude to our supervisors, Dr. Manal Tantawi, and T.A. Manar Sultan, who have supported us throughout the thesis with their patience, knowledge, and experience.

Finally, we would like to thank our friends and all the people who gave us support and encouragement.

# Abstract

Automatic Speech Recognition (ASR) technology has significantly advanced, yet it struggles with accurately transcribing speech from individuals with disorders like stuttering and lisping. SpeCor addresses these limitations by incorporating specialized correction mechanisms for these disorders, in turn enhancing ASR accuracy and communication accessibility. SpeCor uses a minimalist preprocessing approach, focusing on robust training with noise and data augmentation to simulate real-world conditions. Feature extraction relies on Mel-Frequency Cepstral Coefficients (MFCCs) to capture essential sound characteristics (i.e. phonemes). The Whisper model, along with the advanced GPT-2 tokenizer, processes diverse speech patterns. Evaluations using LibriSpeech and LibriStutter datasets show significant improvements for impediment-included speech in Word Error Rate (WER) when training on inclusive datasets (from 95.5% to 16% WER). SpeCor’s architecture, with convolutional layers and transformer blocks, ensures stable and efficient training. This innovation not only enhances the user experience for individuals with speech impediments but also paves the way for further advancements in assistive technologies; this makes voice-based technology more inclusive and reliable for all users. By addressing the unique challenges of speech disorders, SpeCor fosters a more inclusive digital future where everyone can communicate effortlessly using voice-based technologies.

ملخص

لقد تطورت تقنية التعرف التلقائي على الكلام (ASR) بشكل كبير، إلا أنها تواجه صعوبة في نسخ الكلام بدقة من الأفراد الذين يعانون من اضطرابات مثل التأتأة واللثغة. يعالج SpeCor هذه القيود من خلال دمج آليات التصحيح المتخصصة لهذه الاضطرابات، مما يعزز دقة ASR وإمكانية الوصول إلى الاتصالات. تستخدم SpeCor الحد الأدنى من المعالجة المسبقة، مع التركيز على التدريب القوي مع الضوضاء وزيادة البيانات لمحاكاة ظروف العالم الحقيقي. يعتمد استخراج الميزات على معاملات Mel-Frequency Cepstral (MFCCs) لالتقاط خصائص الصوت الأساسية. يعمل جهاز التشفير GPT-2 المتقدم على تحسين التعامل مع الكلمات المعقدة، كما تعمل بنية محول التشفير ووحدة فك التشفير، التي يجسدها نموذج Whisper، على معالجة أنماط الكلام المتنوعة بكفاءة. تُظهر التقييمات باستخدام مجموعات بيانات LibriSpeech و LibriStutter تحسينات كبيرة في معدل أخطاء الكلمات (WER) عند التدريب على مجموعات البيانات الشاملة. تضمن بنية SpeCor ، ذات الطبقات التلافيفية وكتل المحولات، تدريبًا مستقرًا وفعالًا. لا يعزز هذا الابتكار تجربة المستخدم للأفراد الذين يعانون من التأتأة واللثغة فحسب، بل يمهد الطريق أيضًا لمزيد من التقدم في التقنيات المساعدة، مما يجعل التكنولوجيا القائمة على الصوت أكثر شمولاً وموثوقية لجميع المستخدمين. من خلال معالجة التحديات الفريدة لاضطرابات النطق، تعمل SpeCor على تعزيز مستقبل رقمي أكثر شمولاً حيث يمكن للجميع التواصل دون عناء باستخدام التقنيات الصوتية.

Table of Contents

[Acknowledgments III](#_Toc170441840)

[Abstract IV](#_Toc170441841)

[List of Figures VII](#_Toc170441842)

[List of Tables VII](#_Toc170441843)

[1. Chapter One: Introduction 8](#_Toc170441844)

[**1.1.** **Problem Definition** 8](#_Toc170441845)

[**1.2.** **Motivation** 8](#_Toc170441846)

[**1.3.** **Objective** 8](#_Toc170441847)

[**1.4.** **Background** 9](#_Toc170441848)

[**1.5.** **Time Plan** 10](#_Toc170441849)

[**1.6.** **Methodology** 10](#_Toc170441850)

[2. Chapter Two: Literature Review 12](#_Toc170441851)

[**2.1 Theoretical Background** 12](#_Toc170441852)

[**2.2 Related Works** 14](#_Toc170441853)

[**2.2.1. Auto-regressive models (AR Models)** 14](#_Toc170441854)

[**2.2.2. Non-Auto-Regressive Models in NLP** 18](#_Toc170441855)

[**2.3. Observations** 20](#_Toc170441856)

[3. Chapter Three: Architecture and Implementations 23](#_Toc170441857)

[**3.1.** **Model Architecture** 23](#_Toc170441858)

[**3.2.** **Preprocessing** 24](#_Toc170441859)

[**3.3.** **Feature Extraction** 25](#_Toc170441860)

[**3.4.** **Tokenizer for Speech-to-Text Conversion** 26](#_Toc170441861)

[**3.5.** **Model Structure** 27](#_Toc170441862)

[**3.6.** **Inference** 31](#_Toc170441863)

[**3.6.1** **Greedy Decoding** 31](#_Toc170441864)

[**3.6.2**  **Beam Search Decoding** 31](#_Toc170441865)

[4. Chapter Four: Experimental Results 32](#_Toc170441866)

[**4.1.** **Dataset** 32](#_Toc170441867)

[**4.1.1.** **LibriStutter Data Repository** 32](#_Toc170441868)

[**4.1.2.** **The LibriSpeech ASR Dataset** 33](#_Toc170441869)

[**4.2.** **Performance Metric** 36](#_Toc170441870)

[**Word Error Rate (WER)** 36](#_Toc170441871)

[**4.3.** **Tools** 37](#_Toc170441872)

[**4.4.** **Technologies** 37](#_Toc170441873)

[**4.5.** **Results** 39](#_Toc170441874)

[5. Chapter Five: User Manual 41](#_Toc170441875)

[6. Chapter Six: Conclusion & Future Work 50](#_Toc170441876)

[**6.1.** **Conclusion** 50](#_Toc170441877)

[**6.2.** **Future Work** 50](#_Toc170441878)

[**6.3.** **References** 51](#_Toc170441879)

# List of Figures

[**Figure ‎2.1 Whisper Architecture** 15](#_Toc170441971)

[**Figure ‎2.2 BART Architecture** 17](#_Toc170441972)

[**Figure ‎2.3 FastCorrect Model** 18](#_Toc170441973)

[**Figure ‎2.4 SoftCorrect Model** 20](#_Toc170441974)

[**Figure ‎3.1 Model Architecture** 23](#_Toc170441975)

[**Figure ‎3.2 Model Structure** 27](#_Toc170441976)

[**Figure ‎5.1 Landing Page** 41](#_Toc170441977)

[**Figure ‎5.2 Register Account** 42](#_Toc170441978)

[**Figure ‎5.3 User Logging In** 43](#_Toc170441979)

[**Figure ‎5.4 Chat Home Page** 44](#_Toc170441980)

[**Figure ‎5.5 User Adding People into Chat List Using Contacts List** 45](#_Toc170441981)

[**Figure ‎5.6 "Microphone With Checkmark" Button for Conversion of Voice Note into Text or Speech.** 46](#_Toc170441982)

[**Figure ‎5.7 Microphone and Keyboard Input Options for Chat** 46](#_Toc170441983)

[**Figure ‎5.8 User Sending Text Message** 47](#_Toc170441984)

[**Figure ‎5.9 User Choosing Text Option** 48](#_Toc170441985)

[**Figure ‎5.10 User Choosing Speech Option** 49](#_Toc170441986)

# List of Tables

[**Table ‎1.1 Time Plan** 10](#_Toc170441987)

[**Table ‎2.1 Comparative Analysis of Various Techniques Used in Speech Recognition** 13](#_Toc170441988)

[**Table ‎2.2 WER** 16](#_Toc170441989)

[**Table ‎2.3 WER of Multilingual** 16](#_Toc170441990)

[**Table ‎2.4 Error of RoBART** 17](#_Toc170441991)

[**Table ‎2.5 Comparison Between A Typical ASR System and RoBART** 17](#_Toc170441992)

[**Table ‎2.6 Correction Accuracy and Inference Latency of Different Correction Models.** 19](#_Toc170441993)

[**Table ‎2.7 Errors of SoftCorrect Using CER** 20](#_Toc170441994)

[**Table ‎2.8 Summary of Related Works** 21](#_Toc170441995)

[**Table ‎4.1 Results Using WER** 39](#_Toc170441996)

# **Chapter One: Introduction**

## **Problem Definition**

Current ASR systems are designed to handle typical speech patterns, making them inadequate for accurately transcribing speech from individuals with stuttering and lisping. Stuttering is characterized by repetitions, prolongations, and blocks that disrupt the flow of speech, while lisping involves substituting sibilant sounds with acoustically similar but distinct sounds. These distortions significantly affect the acoustic features of speech, leading to poor recognition accuracy. This problem limits the effectiveness of ASR technology for individuals with these speech disorders, restricting their ability to benefit from advancements in voice-based communication.

## **Motivation**

Voice-based communication technologies have become an integral part of our daily lives, used in applications ranging from virtual assistants and transcription services to customer support and accessibility tools. However, these technologies often fall short for individuals with speech impediments such as stuttering and lisping. These speech patterns can lead to significant inaccuracies in transcription, causing frustration and limiting the usability of ASR systems for these users. Our motivation is to create an inclusive ASR system, SpeCor, that accurately transcribes speech for users with these impediments, thus enhancing their ability to interact with technology and access digital services seamlessly.

## **Objective**

The primary objective of SpeCor is to develop an advanced ASR system tailored to the needs of individuals with speech impediments, particularly those who stutter or have lisps. The system aims to:

1. **Speech-to-Text Conversion**: Develop a robust system that effectively converts spoken input into accurate transcription text.
2. **Speech Recognition**: Ensure the system accurately recognizes speech, even when it includes stuttering or lisping.
3. **Real-Time Speech Correction**: Correct speech with lisps and stuttering in real-time to provide smooth and accurate transcriptions.
   1. **Enhance Accessibility and Inclusivity**: Enhance the accessibility and inclusivity of ASR technology by providing accurate transcriptions for individuals with speech impediments.
   2. **Improve Communication**: Improved accuracy in transcribing and reproducing speech enhances the overall quality of communication for individuals with speech impediments and accents. Helps individuals with speech impediments such as lisps or stutters to communicate more effectively, reducing potential barriers and misunderstandings.
4. **Language Learning**: Language learners can benefit from accurate text transcriptions of spoken language, aiding in pronunciation and comprehension.
5. **Educational Tools**: Students with speech impediments can use voice-to-text software more effectively, aiding in note-taking and participation in virtual classrooms.
6. **Employment Opportunities**: Professionals can utilize speech recognition software for tasks such as dictation, customer service interactions, and voice-activated systems without the hindrance of misrecognition.
7. **Content Creation**: Facilitates content creation for podcasts, video captions, and other media by automatically generating text from audio recordings.
8. **Social Interactions**: Enhanced ASR systems facilitate better communication in social media, messaging apps, and other digital platforms, helping individuals maintain personal connections.
9. **Access to Services**: Improved accuracy in voice-activated services, such as virtual assistants and customer support, ensures that individuals with speech impediments can access information and services with ease.

## **Background**

1. **Overview of ASR Technology**

ASR technology has seen tremendous advancements over the past few decades due to improvements in machine learning, neural networks, and computational power. Modern ASR systems can accurately transcribe typical speech patterns and are used in a variety of applications, from virtual assistants like Siri and Alexa to transcription services and customer service tools. However, these systems often fail to accurately transcribe speech that deviates from the norm, such as speech from individuals with disorders like stuttering and lisping.

1. **Challenges with Speech Disorders**

Speech disorders can significantly distort the acoustic features of speech, leading to poor recognition accuracy in standard ASR systems. Stuttering is characterized by repetitions, prolongations, and blocks, while lisping involves the misarticulation of sibilant sounds. These characteristics can confuse ASR systems, resulting in inaccuracies and misunderstandings. Addressing these challenges requires developing specialized algorithms that can recognize and correct these speech patterns.

1. **Importance of Inclusive ASR Systems**

Inclusive ASR systems are crucial for enhancing communication accessibility for individuals with speech disorders. Accurate speech-to-text transcription can significantly improve the ability of these individuals to interact with technology, participate in digital communications, and access services that rely on voice commands. By developing systems like SpeCor, we can make ASR technology more inclusive and reliable, ensuring that it caters to all users, regardless of their speech characteristics.

## **Time Plan**

**Table ‎1.1 Time Plan**

|  |  |  |
| --- | --- | --- |
| **Project Activities** | **Start Date** | **End Date** |
| Survey and Background | 5-10-2023 | 13-10-2023 |
| Dataset Preprocessing | 13-10-2023 | 22-10-2023 |
| Building Models | 23-10-2023 | 28-1-2024 |
| Test and Modify Models | 29-1-2024 | 20-2-2024 |
| Build Application | 21-2-2024 | 27-6-2024 |
| Project Documentation | 13-10-2023 | 27-6-2024 |

## **Methodology**

**Signal Processing**

The first step in developing an accurate ASR system is preprocessing the audio input to enhance the clarity of the speech signal. This involves techniques such as noise reduction, normalization, and filtering. Noise reduction removes background noise that can interfere with speech recognition, while normalization ensures that the audio signal has a consistent volume level. Filtering removes unwanted frequencies that do not contribute to speech recognition, further improving the clarity of the signal.

**Recognition Algorithms**

SpeCor utilizes state-of-the-art speech recognition algorithms that are specifically designed to handle a wide range of speech variations and accents. These algorithms are trained on large datasets that include diverse samples of speech, including those with stuttering and lisps. This ensures that the system can accurately recognize and transcribe speech from individuals with these impediments. The algorithms are integrated into a cohesive system that seamlessly converts speech to text, providing highly accurate transcriptions.

**Correction Mechanisms**

To address the specific challenges posed by stuttering and lisping, SpeCor incorporates advanced correction mechanisms. These mechanisms detect and correct speech patterns specific to these disorders in real time. For example, the system can identify repetitions, prolongations, and blocks associated with stuttering and correct them to provide smooth and accurate transcriptions. Similarly, it can detect misarticulated sibilant sounds associated with lisping and correct them to ensure that the transcription accurately reflects the intended speech.

* 1. **Document Outline:**

**Chapter 2:**

The chapter on related works effectively surveys a wide range of techniques and models in the fields of speech recognition, text-to-speech (TTS), and error correction

**Chapter 3:**

Discusses the speech-to-text system’s architecture and implementation, focusing on a minimalist preprocessing approach, feature extraction, GPT-2 tokenizer, Whisper model architecture, and inference methods. It highlights the use of noise augmentation, Mel-Frequency Cepstral Coefficients, GPT-2 tokenizer, encoder-decoder Transformer, and Key-Value cache for efficient processing.

**Chapter Four:**

Experimental Results shows that ASR systems trained only on LibriSpeech perform well on non-stuttered speech but poorly on stuttered speech, while those trained only on LibriStutter have low overall accuracy. Combining both datasets yields a balanced performance, improving accuracy for stuttered speech and maintaining reasonable accuracy for non-stuttered speech, emphasizing the need for inclusive datasets to develop robust ASR systems.

**Chapter Five:**

Conclusion & Future Work discusses the advancements and challenges in ASR technology, focusing on SpeCor, a system designed to enhance speech recognition accuracy for individuals with speech disorders like stuttering and lisping. It highlights ASR’s evolution, the specific difficulties of recognizing disordered speech, and the extensive benefits of improved ASR systems across various fields. SpeCor employs advanced machine learning and diverse datasets for better accuracy and real-time correction.

**References:**

References for papers and websites used to come out with this project.

# **Chapter Two: Literature Review**

## **2.1 Theoretical Background**

In recent years, automatic speech recognition (ASR) technology has advanced significantly, driven by improvements in machine learning algorithms, increased computational power, and the availability of large datasets. ASR systems have found applications in various domains, including virtual assistants, transcription services, and accessibility tools. Contemporary ASR systems, such as ZIPFORMER, FAIRSEQ S2T, Soft correct, and Fast Correct, utilize sophisticated neural network architectures, particularly deep learning models like Sequence to Sequence, Auto-regressive, Non-Auto Regressive, and Transformer models. These systems are trained on vast amounts of speech data, allowing them to handle various accents, dialects, and noisy environments with remarkable accuracy.

Feature extraction is a crucial step in the development of Automatic Speech Recognition (ASR) systems. It involves transforming raw audio signals into a set of features that can be effectively used by machine learning models to recognize speech. Several feature extraction techniques are commonly used in ASR systems among which are Linear Predictive Coding (LPC), and Mel-Frequency Cepstrum Coefficient (MFCC).

In the following sections, we will discuss the common feature extraction methods and categorizing models based on auto-regression or non-existence of.

**Table ‎2.1 Comparative Analysis of Various Techniques Used in Speech Recognition**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TECHNIQUES** | **FINDINGS** | **ISSUES** |
| **FEATURE**  **EXTRACTION** | Linear Predictive  Coding (LPC) | Static feature extraction method.  Spectral analysis is done with a  Fixed resolution along a subjective frequency scale. | Frequencies are weighted equally on a linear scale  while the frequency sensitivity of the human ear is close to the logarithmic |
| Mel-Frequency  Cestrum Co-efficient  (MFCC) | It is the nearest feature extraction  method to the actual human auditory  speech perception. | MFCC values are not very robust in the presence of additive noises. Normalization is required |
| Dynamic Time  Warping (DTW) | It is used to cope with different speaking speeds. Simple hardware implementation. | Difficulty in selecting the  reference template. |
| **PATTERN**  **MATCHING** | Template Based | Simple Approach Errors due to  segmentation or classification of  smaller acoustically more variable units is avoided. It is speaker-dependent. | The pre-recorded templates are fixed.  Template training and matching become impractical as vocabulary  size increases.  Continuous speech recognition is not possible. |
| Knowledge-based | Uses information regarding linguistics, phonetics, and spectrograms. | Explicit modeling variation in speech is difficult to obtain and use successfully,  so, this approach is impractical. |
| Neural-based | Solve complicated recognition  tasks. Reduces modeling unit. Can be used to develop hybrid models | - |
| Statistical-based | Present models use this approach | Low accuracy of priori modeling presumption reducing its trend |
| Hidden Markov  Model (HMM) | HMMs are simple, automatically trained and computationally feasible to use | Lack in discrimination  property for classification |

Table 2.1 outlines the various techniques used for feature extraction and pattern matching in speech recognition. Each technique presents its own set of advantages and challenges, with modern methods such as neural-based approaches offering significant potential despite their complexities. Understanding these trade-offs is crucial for selecting the appropriate methods for specific speech recognition applications.

## **2.2 Related Works**

### **2.2.1. Auto-regressive models (AR Models)**

**Definition and Principle**

Auto-regressive models in NLP operate on the principle that the probability of a word (or token) in a sequence can be conditioned on the previous words. The model generates text by predicting one word at a time, feeding the predicted word back into the model to predict the next one. This sequential generation captures the dependencies and structure of the language.

**Mathematical Formulation:**

Given a sequence of words w1, w2, …, wT, an auto-regressive model estimates the probability of the sequence as:

(2.1)

Where is the probability of the word given the previous words in the sequence.

Radford et al. [7] present a study on robust speech recognition models trained with large-scale weak supervision. The models, named Whisper, demonstrate strong generalization capabilities, often competitive with fully supervised methods. They approach human accuracy and robustness and are released publicly for further research. Authors critique the brittleness of patterns learned by models trained on specific datasets, which do not generalize well to other distributions.

They highlight the limitations of unsupervised pre-training methods that require fine-tuning and advocate for systems that work reliably across various environments without the need for dataset-specific adjustments.

In Figure 2.1, the Whisper architecture is a simple end-to-end approach, implemented as an encoder-decoder transformer. Input audio is split into 30-second chunks, converted into a log-Mel spectrogram, and then passed into an encoder. A decoder is trained to predict the corresponding text caption, intermixed with special tokens that direct the single model to perform tasks such as language identification, phrase-level timestamps, multilingual speech transcription, and to-English speech A diagram of a block diagram

Description automatically generatedtranslation.

**Figure ‎2.1 Whisper Architecture**

Discussing the correlation between the amount of training data and performance, noting that while performance improves with more data, unique scripts, and linguistic distance can affect transferability.

**Table ‎2.2 WER**

A table with numbers and symbols

Description automatically generated

Whisper models outperform several supervised and commercial systems in long-form transcription tasks and are competitive with human transcription services in terms of accuracy. Whisper models show strong performance in multilingual speech recognition and translation tasks, with the best results observed in medium and low-resource language settings.

**Table ‎2.3 WER of Multilingual**

A table of numbers with black text

Description automatically generated

Dutta et.al. [8]introduce RoBART (Robust BART model), focusing on correcting various types of Automatic Speech Recognition (ASR) errors that frequently appear in the output of a speech recognition system. The table below illustrates an example of ASR output and the corresponding speech, highlighting common speech errors in an ASR system.

They built on the Bidirectional Auto-Regressive Transformer, BART, which is a pre-trained Transformer designed to predict an original text sequence by denoising a given masked and shuffled sequence. However, an off-the-shelf BART model cannot be expected to correct ASR-based errors, as it is trained in a speech-error agnostic setting. To identify potentially common ASR errors that appear regardless of underlying ASR architecture, they use a pre-trained wav2vec2 [25] (Baevskiet et al., 2020) model as a base ASR system to derive predictions for the training data then they train a weaker ASR model using a weaker ASR model allows us to retrieve some ASR errors that are specific to the ASR system.

**A diagram of a train and a train

Description automatically generated**

**Figure ‎2.2 BART Architecture**

RoBART builds upon BART’s foundation by specializing in correcting errors introduced during the ASR process. Through targeted training and adaptation, RoBART aims to improve the accuracy and reliability of transcriptions produced by ASR systems, making it a valuable tool in speech-processing applications.

**Table ‎2.4 Error of RoBART**

|  |  |  |  |
| --- | --- | --- | --- |
| Dataset | Sentences | Tokens | Error% |
| Synthetic (Train) | 2.6M | 45.5M | 5.7 |
| HiWikEd (Test) | 13K | 208K | 6.7 |

**Table ‎2.5 Comparison Between A Typical ASR System and RoBART**

|  |  |  |
| --- | --- | --- |
| Data | ASR | ROBART |
| Test US | 22.40 WER | 19.52 WER |
| Test Non-US | 45.49 WER | 42.39 WER |
| Dev US | 24.90 WER | 21.80 WER |
| Dev Non-US | 36.90 WER | 33.48 WER |

### **2.2.2. Non-Auto-Regressive Models in NLP**

**Definition:**

Non-auto-regressive (NAR) models are a class of models in NLP that generate sequences of words (or tokens) in parallel, rather than one at a time. Unlike auto-regressive models, which predict each word based on the sequence of previously generated words, NAR models attempt to predict multiple tokens simultaneously, often aiming to enhance generation speed and efficiency. These models are particularly useful in applications where latency and computational efficiency are crucial.

**Mathematical Formulation:**

A non-auto-regressive model estimates the probability of the entire sequence without conditioning each word on the previously generated words. One common approach is to assume conditional independence among tokens given some global context c:

(2.2)

There could be a representation of the entire input sequence, such as the output of an encoder in a sequence-to-sequence model.

Leng et.al. [11]proposed NAR error correction for ASR, which greatly reduces the inference latency (up to 9×) compared with its autoregressive counterpart while achieving nearly comparable accuracy. Their method also outperforms the popular NAR models adopted in machine translation and text editions by a large margin. Inspired by the distinctive error patterns and correction operations (i.e., insertion, deletion, and substitution) in ASR, we leverage edit alignments between the output text from ASR models and the ground-truth text to guide the training of NAR error correction, which is critical to FastCorrect

A diagram of a transformer decoder

Description automatically generated

**Figure ‎2.3 FastCorrect Model**

They use a transformer as the basic model architecture of FastCorrect, as shown in the figure above. The encoder takes the source sentence as input and outputs a hidden sequence, which is 1) fed into a length predictor to predict the number of target tokens corresponding to each source token, and 2) used by the decoder through encoder-decoder attention. Figure 2, which is optimized with MSE loss. Thanks to the designs of the edit alignment and length predictor in FastCorrect, the deletion and insertion errors are detected by predicting a length of 0 or more than 1 on the corresponding source token through the length predictor. For substitution errors, the length predicted by the length predictor is 1, which is the same as the length of the unchanged or correct source token. In this condition, the substitution error can be differentiated from the unchanged token by the decoder since it is different with the target token.

**Table ‎2.6 Correction Accuracy and Inference Latency of Different Correction Models.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| AISHELL-1 | Test Set | | Dev set | | Latency (ms/sent) on Test set | | |
| WER | WERR | WER | WERR | NVIDIA P40  GPU | 4-core CPU | single-core CPU |
| No correction | 4.83 | - | 4.46 | - | - | - | - |
| AR model | 4.08 | 15.53 | 3.80 | 14.80 | 149.5(1×) | 248.9 (1×) | 531.3 (1×) |
| LevT(MIter=1) | 4.73 | 2.07 | 4.37 | 2.02 | 54.0 (2.8×) | 82.7 (3.0×) | 158.1 (3.4×) |
| FastCorrect | 4.16 | 13.87 | 3.89 | 13.3 | 21.2 (7.1×) | 40.8 (6.1×) | 82.3 (6.5×) |

Results demonstrate the effectiveness of FastCorrect in speeding up the inference of error correction while maintaining the correction accuracy.

Leng et.al. [6]propose SoftCorrect with a soft error detection mechanism for ASR error correction. They designed a novel language model loss for the encoder to enable error detection and a constrained CTC loss for the decoder to focus on the tokens that are detected as errors.

Error detection can be achieved via the alignments between the target (correct) sentence and the source (incorrect) sentence. Explicit Alignment By explicitly aligning the source and target sequences together with edit distance, we can obtain the number of target tokens (duration) aligned with each source token and train a duration predictor. Thus, we can detect insertion, deletion, and substitution errors with corresponding duration (e.g., 0 stands for deletion error, 1 stands for no change or substitution error, and ≥ 2 stands for insertion error).

However, the duration predictor is hard to optimize precisely, and thus new error will be introduced once duration prediction is not accurate. Implicit alignment errors can be “detected” via implicit alignment between target and source sequence.

A diagram of a computer

Description automatically generated

**Figure ‎2.4 SoftCorrect Model**

The figure above is an overview of SoftCorrect. We use A B C E to represent the ground-truth tokens, while B′ B′′ B′′′ D′ E′ to represent incorrect tokens. We use ϕ to represent blank tokens for alignment purposes only, which is leveraged in both multi-candidate alignment and CTC alignment. In this case, the ground-truth sentence is ABCE, while the 4 candidates are AB′CD′E, AB′CD′E′, AB′′CD′E′, and AB′′′D′E, respectively. The selected candidate is AB′′CD′E, where B′′ and D′ are detected as incorrect tokens and duplicated when fed into the decoder.

**Table ‎2.7 Errors of SoftCorrect Using CER**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | AISHEL – 1 | | Aidatang | |
| Test | Dev | Test | Dev |
| SoftCorrect | 3.57 | 3.4 | 4.05 | 3.44 |

Thus, in this paper, they adopt the CTC-based solution but enhance it with a soft error detection mechanism

## **2.3. Observations**

LibriSpeech provides approximately 1000 hours of 16kHz English speech derived from read audiobooks, meticulously segmented and aligned from the LibriVox project. However, it lacks natural stuttering, limiting its utility for training models focused on stuttered speech recognition.

In contrast, the LibriStutter dataset synthesizes stuttered speech from the LibriSpeech corpus, offering time-aligned transcriptions and labels for five types of artificial stuttering. Despite its detailed classification of stutter types, its small size and artificial nature pose challenges for training robust stuttered speech recognition models.

The Common Voice Corpus stands out for its vast, publicly available voice data, fostering innovation and competition in speech technology. However, its flawlessly pronounced words without natural stuttering or lisps make it unsuitable for training models that need to recognize speech disorders like stuttering.

MuST-C, a multilingual speech translation corpus, provides paired English-Spanish data, requiring substantial computational resources for processing and utilization in multilingual speech technology applications.

WenetSpeech offers a multi-domain Mandarin speech corpus exceeding 10,000 hours, derived from YouTube and podcasts. Despite its extensive size and innovative error detection methods, its Mandarin language focus limits direct applicability to models targeting other languages.

Aishell-1 and Aidatatang are Mandarin speech corpora characterized by high transcription accuracy and diverse regional accents, suitable for Mandarin-specific speech recognition models due to their language specificity and rigorous data quality standards.

Manually collected data used in specific research papers offer tailored datasets but are costly and time-consuming to acquire, posing accessibility challenges for broader research and development efforts in speech recognition.

**A white rectangular grid with black text

Description automatically generatedTable ‎2.8 Summary of Related Works**

A white sheet with black text

Description automatically generatedTo summarize the following table, the most popular datasets used are LibriSpeech, Common Voice Corpus, Wall Street Journal, and Aishell-1.

Most of the papers are void of preprocessing techniques, but the most frequent besides feature extraction are Data Augmentation and Data Normalization.

The Most Commonly Used Feature Extraction Technique is Mel-Frequency Cepstral Coefficients.

The Suitable Evaluation Metrics for Speech Recognition are Word Error Rate and Character Error Rate.

Most recent advancements in Automatic Speech Recognition (ASR) prominently feature transformer architectures and attention mechanisms, renowned for their ability to achieve high accuracy. These models leverage transformer frameworks to effectively capture dependencies across longer sequences of speech data, enhancing both the robustness and precision of ASR systems.

# **Chapter Three: Architecture and Implementations**

## **Model Architecture**

A diagram of a computer program

Description automatically generated

**Figure ‎3.1 Model Architecture**

The architecture of SpeCor is designed to enhance Automatic Speech Recognition (ASR) for individuals with speech disorders, specifically stuttering and lisping. The input is spoken language, which undergoes minimalist preprocessing, including noise introduction and data augmentation, to simulate real-world conditions. This preprocessing reduces the need for extensive noise removal and improves real-time transcription efficiency. The system extracts features using Mel-Frequency Cepstral Coefficients (MFCCs), capturing essential sound characteristics after steps like pre-emphasis, windowing, FFT, and DCT. The GPT-2 tokenizer then breaks down the audio into sub words or characters, enhancing the model’s handling of complex words. SpeCor employs an encoder-decoder Transformer architecture, exemplified by the Whisper model, to process these features. The encoder uses convolutional layers and Transformer blocks with sinusoidal position embeddings, while the decoder, equipped with learned position embeddings and cross-attention layers, generates the output sequence. This setup includes pre-activation residual connections and layer normalization for training stability and efficiency, along with a Key-Value (KV) cache for faster inference. During inference, the system processes new audio data to generate text tokens, using methods like greedy decoding or beam search to ensure accuracy. The output is highly accurate transcriptions that closely match the spoken input, even for speech with stuttering or lisping, ensuring clarity and reliability in the converted text. This comprehensive approach results in improved ASR performance, making voice-based technology more inclusive and accessible.

In the following sections, we will go in-depth on each part of the model architecture, the preprocessing, and feature extraction techniques that have been used in details.

## **Preprocessing**

In recent times, there has been a trend towards minimalist preprocessing for speech-to-text conversion systems. This approach focuses on enhancing the model's ability to handle real-world conditions by intentionally introducing noise during the training phase and leveraging data augmentation techniques.

* By adding noise to the data to challenge the model to handle it and provide a more realistic representation of the subject to simulate the real word voice.
* To provide real-time Speech-To-Text conversion, by reducing the time needed to remove background noise.
* In real-life conversations, it is very rare for there to be no noise, and the sound is rarely 100% clear.
* Data augmentation is used to improve training, that enforces the model to detect the right transcribe in real situations

**Minimalist Preprocessing Approach:**

* **Purpose**: To challenge the model to handle noisy environments, provide a realistic representation of real-world voice conditions, and improve the robustness and generalization of the model.
* **Benefits**: This approach helps make the model robust to various types of noise, ensuring it performs well in real-life scenarios where background noise is common. It also reduces the preprocessing time needed for noise removal, enabling efficient real-time transcription. Furthermore, it enforces the model to learn to detect correct transcriptions under different conditions, improving its performance in real-world situations.
* **Techniques**:
  1. **Adding Noise to Data**: Artificial noise (e.g., white noise, background chatter, traffic sounds) is added to the audio data. For example, recordings of common environmental noises are superimposed on the speech data to simulate a realistic audio environment.
  2. **Real-Time Training on Noisy Data**: Models are trained on noisy data directly, ensuring they learn to extract relevant speech features even in the presence of noise. This prepares the models to handle the background noise typically present in real-life conversations.
  3. **Data Augmentation**: Various types of background noises are added at different intensity levels. This exposure to diverse acoustic environments ensures that the model generalizes well to different types of noise and conditions.

## **Feature Extraction**

**Mel-Frequency Cepstral Coefficients (MFCC)**

The process of computing MFCC involves several steps, each crucial for transforming the raw audio signal into a set of meaningful features that can be used for tasks like speech recognition, speaker identification, and more.

1. **Pre-emphasis**:
   * **Purpose**: This step is used to improve the signal quality at the output of a data transmission by boosting the energy of the high frequencies.
2. **Windowing**:
   * **Purpose**: Used to normalize the signal quality by applying a window to the signal, such as a Hamming window, to reduce spectral leakage.
   * **Description**: This windowing helps in minimizing the discontinuities at the boundaries of each frame.
3. **Fast Fourier Transform (FFT)**:
   * **Purpose**: Converts the signal from the time domain to the frequency domain, which can be used to analyze and remove noise.
   * **Description**: Each windowed frame undergoes an FFT to transform it into the frequency domain. This results in a spectrum or power spectrum of the signal, providing insight into the frequency components present in each frame.
4. **Log Mel Filter Bank**:
   * **Purpose**: Mimics the human auditory system's frequency resolution by applying a log scale to the frequency domain data and passing it through a bank of filters spaced evenly on the mel scale.
5. **Discrete Cosine Transform (DCT)**:
   * **Purpose**: Compresses the features into fewer components with high power, thus reducing redundancy.
   * **Description**: The logarithm of the mel-filtered spectrum is transformed using DCT. This step helps in decorrelating the features and concentrating the energy in the lower dimensions. Typically, the first few coefficients are retained as they contain most of the signal's information. These coefficients are the MFCCs.

By following these steps, the raw audio signal is transformed into a set of MFCCs, which are robust features that capture the essential characteristics of the sound. These features can then be used in various machine learning and signal processing applications to achieve tasks like speech and speaker recognition with high accuracy.

## **Tokenizer for Speech-to-Text Conversion**

**What is a Tokenizer?**

A tokenizer is a tool used in natural language processing (NLP) to convert text into a sequence of tokens. Tokens are smaller units like words, subwords, or characters that the model can process. The process of tokenization helps in breaking down a large text into manageable pieces, making it easier for the model to analyze and understand the input data.

**Purpose of a Tokenizer**

The main purpose of a tokenizer is to prepare text data for model training and inference. By converting text into tokens, the tokenizer enables the model to handle language in a structured manner. This process is essential for various NLP tasks, including speech-to-text conversion, as it helps in capturing the linguistic features and patterns present in the text.

**Benefits of the GPT-2 Tokenizer**

The GPT-2 tokenizer, developed by OpenAI, is known for its rich token representation. Here are some of the key benefits of using the GPT-2 tokenizer:

1. **Subword Tokenization**: The GPT-2 tokenizer uses byte pair encoding (BPE) to split words into subwords or character segments. This allows the tokenizer to handle rare or out-of-vocabulary words by breaking them down into more common subunits, enhancing the model's ability to understand and process diverse vocabulary.
2. **Contextual Understanding**: The GPT-2 tokenizer captures linguistic nuances and contextual information, which is crucial for understanding the meaning and intent behind the text. This capability improves the accuracy and reliability of the model's predictions.
3. **Handling Complex Structures**: The tokenizer effectively manages complex linguistic structures, such as idioms, compound words, and multi-word expressions. This ensures that the model can process and interpret a wide range of text inputs accurately.
4. **Efficiency and Performance**: By using subword tokenization and capturing contextual information, the GPT-2 tokenizer contributes to more efficient model training and inference. It helps in reducing the size of the input sequence while preserving the semantic meaning, leading to faster and more effective processing.

**Why We Chose the GPT-2 Tokenizer**

We chose the GPT-2 tokenizer for our speech-to-text conversion system due to its advanced tokenization capabilities and the benefits it offers:

* **Rich Representation**: The ability to represent text in a rich and nuanced manner ensures that the model can accurately capture the intricacies of human language.
* **Robustness**: The GPT-2 tokenizer's handling of rare and complex words makes the system more robust and versatile, capable of dealing with various linguistic phenomena encountered in real-world scenarios.
* **Enhanced Accuracy**: The contextual understanding provided by the GPT-2 tokenizer enhances the accuracy of the speech-to-text conversion, ensuring that transcriptions are more precise and reflective of the original speech.

By leveraging the sophisticated tokenization process of GPT-2, our speech-to-text system benefits from improved handling of diverse vocabulary and complex linguistic structures, leading to high-quality and reliable transcriptions

## **Model Structure**

A diagram of a diagram

Description automatically generated with medium confidence

**Figure ‎3.2 Model Structure**

Whisper is a general-purpose speech recognition model. It is trained on a large dataset of diverse audio and is also a multitasking model that can perform multilingual speech recognition, speech translation, and language identification.

A Transformer sequence-to-sequence model is trained on various speech-processing tasks, including speech recognition. These tasks are jointly represented as a sequence of tokens to be predicted by the decoder, allowing a single model to replace many stages of a traditional speech-processing pipeline. The multitask training format uses a set of special tokens that serve as task specifiers or classification targets.

Since the focus of our work is on studying the capabilities of large-scale supervised pre-training for speech recognition, we use an off-the-shelf architecture to avoid confounding our findings with model improvements. We chose an encoder-decoder Transformer as this architecture has been well-validated to scale reliably. All audio is re-sampled to 16,000 Hz, and an 80-channel log-magnitude Mel spectrogram representation is computed on 25 millisecond windows with a stride of 10 milliseconds.

For feature normalization, we globally scale the input to be between -1 and 1 with approximately zero mean across the pre-training dataset. The encoder processes this input representation with a small stem consisting of two convolution layers with a filter width of 3 and the GELU activation function where the second convolution layer has a stride of two. Sinusoidal position embeddings are then added to the output of the stem after which the encoder Transformer blocks are applied. The transformer uses pre-activation residual blocks, and a final layer normalization is applied to the encoder output. The decoder uses learned position embeddings and tied input-output token representations. The encoder and decoder have the same width and number of transformer blocks.

Figure 3.1 summarizes the model architecture.

1. **Encoder**
2. **Convolutional Stem**
   * **Specification**: The stem consists of two convolutional layers with a filter width of 3. The first convolution uses a stride of 1, and the second uses a stride of 2. Both layers use the GELU activation function.
   * **Reason**: The convolutional layers act as initial feature extractors, capturing local patterns and reducing the temporal dimension of the input.
   * **Benefit**: Convolutions are effective at detecting local features such as phonemes and small patterns in the audio signal. The reduction in temporal dimension (due to the stride of 2 in the second convolution) makes the subsequent Transformer layers more computationally manageable, while GELU activation provides smoother and potentially more effective non-linearity compared to ReLU.
3. **Sinusoidal Position Embeddings**
   * **Specification**: Sinusoidal position embeddings are added to the output of the convolutional stem.
   * **Reason**: Transformers do not inherently understand the order of input sequences. Position embeddings provide a way to encode positional information.
   * **Benefit**: Sinusoidal embeddings, as opposed to learned embeddings, ensure that the model can generalize to sequence lengths not seen during training. This is crucial for speech recognition tasks, where the length of input sequences can vary significantly.
4. **Transformer Blocks**
   * **Specification**: The encoder uses pre-activation residual blocks with multi-head self-attention and feed-forward layers, followed by layer normalization.
   * **Reason**: Transformer blocks are adept at capturing long-range dependencies in the data, which is essential for understanding the context in speech.
   * **Benefit**: Pre-activation residual blocks improve gradient flow during training, making it easier to train deep networks. The multi-head self-attention mechanism allows the model to focus on different parts of the input sequence simultaneously, capturing intricate dependencies and relationships within the data.
5. **Decoder**
6. **Learned Position Embeddings**
   * **Specification**: Learned position embeddings are added to the input tokens of the decoder.
   * **Reason**: Position embeddings are necessary for the decoder to understand the order of the tokens it is generating.
   * **Benefit**: Learned embeddings allow the model to adapt positional encoding specifically for the task at hand, which can improve performance in generating the output sequence.
7. **Transformer Blocks**
   * **Specification**: The decoder mirrors the encoder, using the same width and number of blocks, but includes cross-attention layers to attend to the encoder’s output.
   * **Reason**: Symmetric architecture between encoder and decoder ensures consistency and simplifies the model design.
   * **Benefit**: The cross-attention layers enable the decoder to focus on relevant parts of the encoded input sequence, facilitating accurate sequence generation. Consistent architecture across the encoder and decoder also simplifies the training and tuning process.
8. **Additional Considerations**

* **Layer Normalization**

**Specification**: Applied after each Transformer block in both encoder and decoder.

**Reason**: Layer normalization stabilizes the learning process by normalizing the inputs to each layer, addressing internal covariate shifts.

**Benefit**: This technique helps in faster convergence and improved training stability, particularly in deep networks like Transformers.

* **Pre-activation Residual Connections**

**Specification**: Used within Transformer blocks.

**Reason**: Residual connections allow gradients to flow directly through the network, mitigating the vanishing gradient problem.

**Benefit**: They enable the training of very deep networks by preserving the gradient magnitude, which is crucial for effective learning in complex architectures.

1. **Key-Value (KV) Cache**

In Whisper, OpenAI's speech recognition model, the Key-Value (KV) cache plays a crucial role in improving the efficiency and speed of processing during inference. Let's dive into what the KV cache is, how it is used in Whisper, its benefits, and its specific implementation details.

* **What is a KV Cache?**
  + - A Key-Value (KV) cache in the context of Whisper, an encoder-decoder Transformer model for speech recognition, is used to store the key and value vectors computed during the self-attention mechanism. These vectors are reused across multiple steps of the decoding process, significantly reducing the need for redundant computations.
* **Why Use a KV Cache in Whisper?**
  + - Whisper processes audio sequences to generate text output. During the decoding phase, each token's generation depends on all previously generated tokens. This dependency is computationally expensive because the self-attention mechanism must repeatedly process the same information for each step of the sequence generation.
* **Efficiency in Long Sequences, and Benefits of Using a KV Cache:**

1. **Reducing Redundancy**

Without a KV cache, the model would recompute the key and value vectors for all previous tokens at each decoding step. By caching these vectors, Whisper avoids redundancy, making decoding more efficient.

1. **Speeding Up Inference**

The KV cache allows Whisper to generate text tokens faster by reusing previously computed key and value vectors, thus significantly speeding up the inference process.

1. **Improved Efficiency**

Less Redundant Computation: The model does not need to recompute key and value vectors for previously processed tokens.

Faster Inference: The reuse of cached vectors accelerates the decoding process, making model responsive and suitable for real applications.

1. **Enhanced Scalability**

Handling Long Sequences: Whisper can efficiently manage longer audio sequences by leveraging the KV cache, which reduces the computational load per decoding step.

1. **Reduced Memory Footprint**

Optimized Memory Usage: Although caching increases memory usage, it is managed efficiently to balance memory and computational resources.

## **Inference**

Inference in machine learning is the phase where a trained model is used to make predictions on new, unseen data. In the context of a speech-to-text model like Whisper, inference involves several steps. First, the audio input is captured and pre-processed, converting the raw audio signal into a set of meaningful features, such as Mel-Frequency Cepstral Coefficients (MFCCs). These features are then fed into the model's encoder, which transforms them into a high-level representation. The decoder processes this representation to generate a sequence of text tokens. During this decoding phase, methods like greedy decoding (choosing the most probable next token at each step) or beam search (considering multiple possible sequences to find the most likely one) are used to form the final text output. The result is a transcription of the spoken input, which can then undergo further post-processing to enhance accuracy and readability. This entire process allows the model to convert spoken language into written text efficiently and effectively. During inference in speech-to-text models like Whisper, you can use either **greedy decoding** or **beam search** to generate text from the model’s predictions, where each method will be discussed:

### **Greedy Decoding**

Greedy decoding is a straightforward decoding strategy where, at each step, the model selects the token with the highest probability as the next token in the sequence.

**Advantages:**

* Simplicity: Easy to implement and understand.
* Speed: Faster than more complex methods like beam search.
* Lower Computational Cost: Requires less computational power and memory.

**Disadvantages:**

* Suboptimal Solutions: Can miss better sequences since it doesn't consider alternative paths.
* Poor in Ambiguous Contexts: May not perform well when multiple tokens have similar probabilities.

### **3.6.2 Beam Search Decoding**

Beam search is a more sophisticated decoding strategy that considers multiple sequences at each step, keeping track of the most promising ones.

**Advantages:**

* Better Quality: Typically produces higher-quality sequences compared to greedy decoding.
* Exploration: Considers multiple possible sequences, reducing the likelihood of missing the optimal sequence.
* Handles Ambiguity: Better at handling situations where multiple tokens have similar probabilities.

**Disadvantages:**

* Complexity: More complex to implement and understand.
* Slower: Computationally more expensive and slower than greedy decoding.
* Higher Memory Usage: Requires more memory to keep track of multiple sequences.

# **Chapter Four: Experimental Results**

## **Dataset**

### **LibriStutter Data Repository**

1. **Introduction**

The **LibriStutter Data** repository is an innovative project designed to aid in stuttering research by providing a dataset with artificial stuttering patterns. The dataset is derived from the well-known LibriSpeech corpus, a staple in the speech recognition community. LibriStutter Data aims to facilitate the development and evaluation of speech recognition systems that can handle speech disfluencies, particularly stuttering. This document provides a detailed explanation of the repository, its structure, the data it contains, and its potential applications.

1. **Repository Structure**
2. **Data Preparation Scripts**:
   * These scripts are the heart of the repository, introducing stuttering patterns into the LibriSpeech dataset.
   * They allow for the simulation of various stuttering behaviors, such as repetitions, prolongations, and blocks.
   * The scripts are designed to be flexible, offering parameters that can be adjusted to control the type and frequency of stuttering patterns.
3. **Documentation**:
   * Comprehensive instructions guide users through the process of setting up their environment, running the scripts, and interpreting the results.
   * It includes installation steps for necessary dependencies and a detailed explanation of script parameters.
4. **Sample Data**:
   * A small subset of the modified dataset is included to demonstrate the stuttering modifications.
   * These samples help users verify that the scripts are functioning correctly and provide an example of what to expect from the full dataset.
5. **Data Description**

The original LibriSpeech dataset is a large corpus of read English speech, primarily sourced from audiobooks. It is widely used in the field of speech recognition due to its high-quality recordings and extensive size. LibriSpeech includes:

* Approximately 1,000 hours of audio.
* Diverse speaker demographics (age, gender, accent).
* Accurate transcriptions accompanying the audio files.

1. **Stuttering Modifications**

The LibriStutter Data enriches the LibriSpeech corpus by integrating realistic stuttering patterns into its audio files. These modifications are strategically designed to simulate common stuttering behaviors such as repetitions, prolongations, and blocks. By embedding these speech disfluencies into the dataset, LibriStutter enhances its utility for training and evaluating speech recognition models that need to accurately interpret and transcribe speech affected by stuttering. Each audio file in WAV format is paired with corresponding transcriptions that detail the stuttering patterns, alongside metadata providing insights into the frequency and types of disfluencies present. This structured approach facilitates comprehensive analysis and development of ASR systems capable of handling diverse speech impediments effectively.

1. **Applications**
2. **Speech Recognition Research**:
   * Developing and testing models that are robust to speech disfluencies.
   * Enhancing the accuracy of speech recognition systems for users with stuttering.
3. **Linguistic Studies**:
   * Analysing the impact of stuttering on speech processing and comprehension.
   * Studying the characteristics of stuttering patterns and their variations.

### **The LibriSpeech ASR Dataset**

1. **Introduction**

The **LibriSpeech ASR** dataset is a cornerstone in the field of Automatic Speech Recognition (ASR). Developed as a high-quality, large-scale corpus of read English speech, LibriSpeech has become a standard benchmark for ASR systems. This dataset, sourced from audiobooks, provides a wealth of speech data suitable for training, evaluating, and benchmarking ASR models. In this document, we will explore the origins, structure, contents, and applications of the LibriSpeech ASR dataset.

1. **Origins and Purpose**

The LibriSpeech dataset, introduced in 2015 by Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur in their paper titled "Librispeech: An ASR corpus based on public domain audiobooks," serves as a pivotal resource for the automatic speech recognition (ASR) research community. This dataset was meticulously curated to meet several critical needs within the field:

It offers a vast repository of speech data, comprising approximately 1,000 hours of recordings, which is essential for training and evaluating robust ASR models. This large-scale availability ensures that researchers have ample data to develop systems capable of handling diverse linguistic patterns and acoustic environments.

The audio recordings in the LibriSpeech dataset are characterized by their high quality, featuring clear speech with minimal background noise and high fidelity. This fidelity is crucial for accurate transcription and ensures that the dataset reflects real-world speech conditions as closely as possible.

1. **Dataset Structure**

The LibriSpeech dataset is meticulously organized to facilitate ease of use. It is divided into multiple parts based on the amount of training data and the difficulty of the test sets.

1. **Training Sets**:
   * **train-clean-100**: 100 hours of clean speech.
   * **train-clean-360**: 360 hours of clean speech.
   * **train-other-500**: 500 hours of speech with more challenging acoustic conditions.
2. **Development Sets**:
   * **dev-clean**: A clean development set for validation purposes.
   * **dev-other**: A development set with more challenging conditions.
3. **Test Sets**:
   * **test-clean**: A clean test set for final evaluation.
   * **test-other**: A test set more challenging conditions for evaluation.

**Additional Metadata:**

* Each audio file is accompanied by a transcription in plain text format.
* Metadata includes details about the speaker, such as gender, age, and the unique identifier.

1. **Data Collection and Preprocessing**

The audio files in LibriSpeech are sourced from the LibriVox project, which provides free audiobooks recorded by volunteers. The selection process ensures that only high-quality recordings are included. The preprocessing steps include:

* Segmentation: Splitting long audiobook recordings into shorter, manageable segments.
* Normalization: Ensuring consistent audio quality and transcription accuracy.
* Quality Control: Reviewing and filtering out segments with poor audio quality or transcription errors.
* Format: WAV
* Sample Rate: 16 kHz
* Bit Depth: 16-bit

1. **Applications**

The LibriSpeech ASR dataset has numerous applications across various domains, primarily in speech recognition and related fields.

1. **Automatic Speech Recognition (ASR)**:
   * The large volume and high quality of the dataset make it ideal for training ASR models.
   * LibriSpeech serves as a standard benchmark for evaluating the performance of ASR systems.
2. **Speaker Recognition**:
   * Using the dataset to train models that identify or verify speakers.
   * Segmenting audio into parts corresponding to different speakers.
3. **Linguistic Research**:
   * Analysing speech patterns, pronunciation variations, and other phonetic aspects.
   * Developing language models that predict the likelihood of sequences of words.
4. **Challenges and Limitations**

While LibriSpeech is an invaluable resource, it does come with certain challenges and limitations:

* Being sourced from audiobooks, the dataset primarily represents read speech, which may not generalize to spontaneous conversational speech.
* Although diverse, the dataset may not fully capture the wide range of accents and dialects present in English-speaking populations globally.
* The "other" subsets introduce variability in acoustic conditions, but real-world environments can be even more challenging.

## **Performance Metric**

## **Word Error Rate (WER)**

Word Error Rate is a valuable metric for assessing and comparing the performance of ASR systems. It is calculated using the following formula:

(4.1)

Where:

* **S:** the number of substitution errors (words that were recognized incorrectly).
* **D:** the number of deletion errors (words from the reference that were not recognized).
* **I**: the number of insertion errors (extra words that were incorrectly added in the recognition).
* **N:** the total number of words in the reference transcription.

WER provides a quantitative measure of the accuracy of an ASR system by considering all types of errors (substitutions, deletions, and insertions) in the recognized text compared to a reference transcription.

## **Tools**

1. **os**: Provides functions for interacting with the operating system, like file and directory manipulation.
2. **warnings**: Allows the developer to control the issuance and handling of warning messages.
3. **soundfile**: Used for reading and writing sound files.
4. **random**: Implements pseudo-random number generators for various distributions and operations.
5. **librosa**: A library for audio and music analysis.
6. **functools**: Provides higher-order functions for functional programming, like caching and partial function application.
7. **subprocess**: Allows spawning new processes, connecting to their input/output/error pipes, and obtaining their return codes.
8. **typing**: Supports type hints for static type checking in Python.
9. **shutil**: Provides high-level operations on files and collections of files.
10. **base64**: Provides functions for encoding binary data to base64 and decoding base64 data to binary.
11. **gzip**: Provides simple compression and decompression of files using the GZIP format.
12. **dataclasses**: A module that provides a decorator and functions for automatically adding special methods to user-defined classes.
13. **tiktoken**: A package for tokenizing text.
14. **more\_itertools**: Extends Python’s itertools with more routines for operating on iterables.
15. **kaggle**: Provides an interface to Kaggle’s API for downloading datasets and interacting with competitions.
16. **elevenlabs**: A client library for ElevenLabs' API, providing text-to-speech and other voice-related functionalities.
17. **pydub**: A library for manipulating audio with a simple high-level interface.
18. **re**: Provides support for regular expressions in Python.
19. **unicodedata**: Provides access to the Unicode Character Database, defining character properties for all Unicode characters.
20. **regex**: An alternative to Python’s built-in re module, offering additional functionalities for regular expressions.
21. **argparse**: A module for parsing command-line arguments.
22. **traceback**: Provides utilities to extract, format, and print stack traces of Python programs.
23. **tqdm**: A fast, extensible progress bar for loops and other iterative tasks.

## **Technologies**

1. **numpy**: A library for numerical computing, offering support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions.
2. **pandas**: A data analysis and manipulation library that provides data structures like DataFrame for handling tabular data.
3. **torch**: The core library of PyTorch for tensor computations and neural network operations.
4. **torch.nn.functional**: Contains functions for neural network operations like activation functions and loss functions.
5. **Flutter**
   * **Description**: An open-source UI toolkit by Google for building cross-platform applications.
   * **Key Features**:
     + Single codebase for Android, iOS, and web
     + Rich set of widgets
     + Hot reload for real-time updates
     + High performance with native ARM code
     + Uses Dart programming language
6. **Firebase**
   * **Description**: A platform by Google for mobile and web app development.
   * **Key Features**:
     + Realtime Database and Cloud Firestore for data storage
     + Authentication services
     + Fast and secure hosting
     + Cloud Functions for backend logic
     + Analytics for user insights
7. **FastAPI**
   * **Description**: A high-performance web framework for building APIs with Python 3.7+.
   * **Key Features**:
     + High performance and asynchronous support
     + Automatic interactive documentation
     + Type safety with Python type hints
     + Dependency injection
     + Production-ready design with comprehensive testing and validation

## **Results**

The evaluation of ASR systems was conducted using two datasets: LibriSpeech and LibriStutter. These datasets were used to train and test the models, providing insights into their performance with both non-stuttered and stuttered speech.

**Table ‎4.1 Results Using WER**

|  |  |  |
| --- | --- | --- |
| **Train Test** | **LibriSpeech** | **LibriStutter** |
| **Using LibriSpeech Only** | **8.3** | **95.5** |
| **Using LibriStutter Only** | **445.39** | **212.05** |
| **Using LibriSpeech & LibriStutter** | **9.24** | **16** |

Using the LibriSpeech dataset alone resulted in a Word Error Rate (WER) of 8.3, showcasing its superior accuracy in recognizing non-stuttered speech. This dataset is well-suited for general Automatic Speech Recognition (ASR) tasks where fluent speech predominates. In contrast, employing only the LibriStutter dataset yielded a significantly higher WER of 445.39, indicating poor performance in recognizing both stuttered and non-stuttered speech patterns. The model trained exclusively on LibriStutter shows limitations in handling diverse speech conditions effectively. However, combining both datasets in training produced a WER of 9.24, demonstrating improved accuracy in recognizing stuttered speech while maintaining reasonable performance for fluent speech.

Evaluation on the LibriStutter dataset using only LibriSpeech data revealed a WER of 95.5, emphasizing a substantial accuracy gap between stuttered and non-stuttered speech. This disparity may lead to potential discrimination against individuals with speech impediments when utilizing ASR systems trained solely on fluent speech data. Conversely, training exclusively on LibriStutter resulted in a WER of 212.05, indicating a bias towards recognizing stuttered speech patterns but achieving low overall accuracy. By integrating both datasets, the combined approach achieved a WER of 16, demonstrating desirable outcomes for both speech types and highlighting the advantages of comprehensive training data.

Comparing training approaches, we notice that, training using LibriSpeech only, achieved the best accuracy for non-stuttered speech. The model is well-suited for general ASR tasks where stuttered speech is not common. However, the model exhibits a significant disparity in performance when tested on stuttered speech, which could lead to bias against individuals with speech impediments.

By training using LibriStutter only, it achieved the worst accuracy for both stuttered and non-stuttered speech. The model is not well-trained for general ASR tasks. Shows a bias towards recognizing stuttered speech but fails to achieve high accuracy overall.

By training using both LibriSpeech and LibriStutter, by combining both datasets and fine-tuning the hyperparameters during training, the model achieved a balance in accuracy for both stuttered and non-stuttered. Although the accuracy decreased for the LibriSpeech dataset, this trade-off resulted in improved performance for stuttered speech. The combined training approach provides a more inclusive ASR system capable of handling a wider range of speech patterns.

The evaluation of ASR systems using Word Error Rate (WER) highlights the importance of considering both non-stuttered and stuttered speech in training datasets. The results demonstrate that training solely on non-stuttered speech (LibriSpeech) leads to high accuracy for typical speech but fails significantly with stuttered speech, training solely on stuttered speech (LibriStutter) results in poor overall performance, indicating the need for a more balanced approach, however combining both datasets and fine-tuning the model improves accuracy for stuttered speech while maintaining reasonable performance for non-stuttered speech.

The findings underscore the necessity of inclusive datasets in developing robust ASR systems. By incorporating diverse speech patterns, ASR systems can be made more equitable and effective for all users, including those with speech impediments.

# **Chapter Five: User Manual**

This chapter is a walkthrough of the whole application.

**5.1 Landing Page**

* This is the first page to be shown after launching the mobile application as shown in Figure 5.1.
* It contains fields for the user to enter their credentials to log in to the application.
* It also contains an option for registering a new user via entering their email.

**A screenshot of a phone

Description automatically generated**

**Figure ‎5.1 Landing Page**

**5.2 User Registration**

* The user taps on “Register” so, they can register via email.
* A registration page will appear requiring the user to fill in the needed data i.e., name, email, password, and image. The user then proceeds by tapping the “Register” button to enter the home page.

A screenshot of a phone

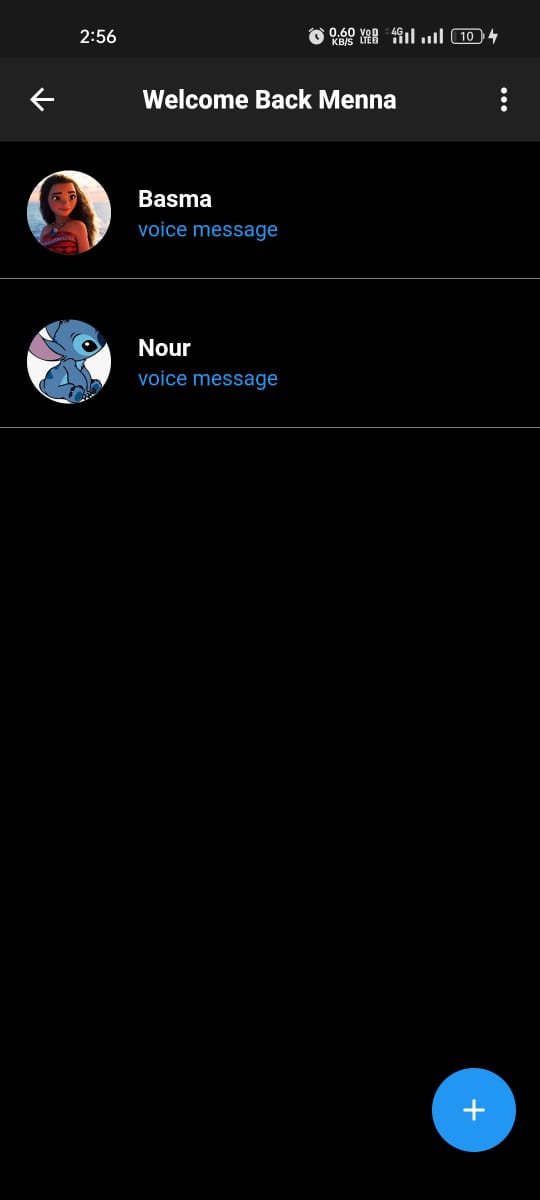
Description automatically generated**A screenshot of a phone

Description automatically generated**

**Figure ‎5.2 Register Account**

**5.3 User Login**

* The user enters their email and password in their respective fields and presses the “Login” button as shown.
* If the email and password are written correctly user will be redirected to the home page.

A screenshot of a phone

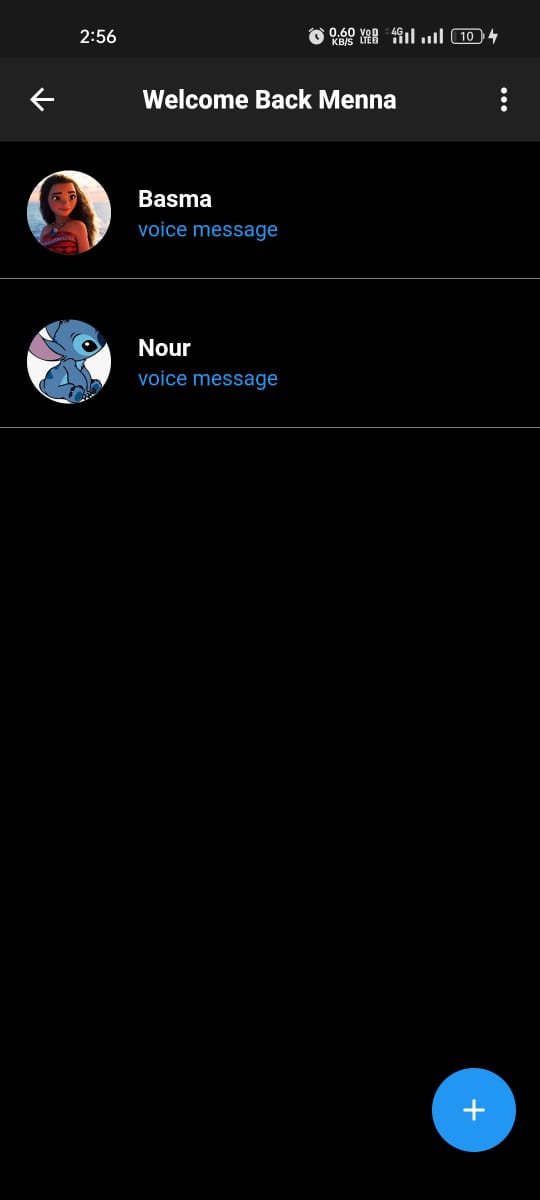
Description automatically generatedA screenshot of a phone

Description automatically generated

**Figure ‎5.3 User Logging In**

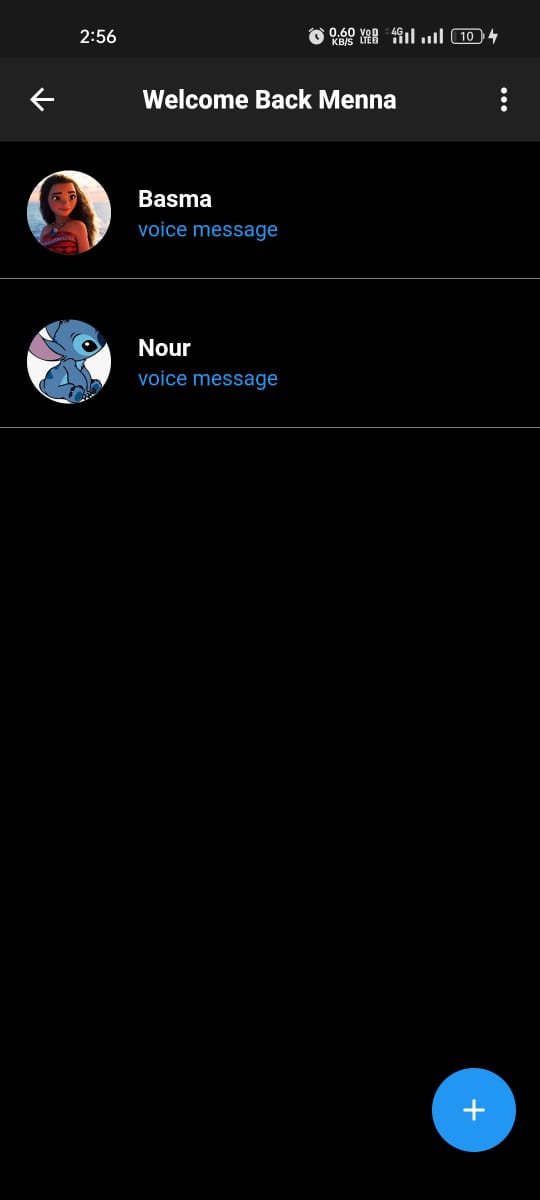
**5.4 Home Page**

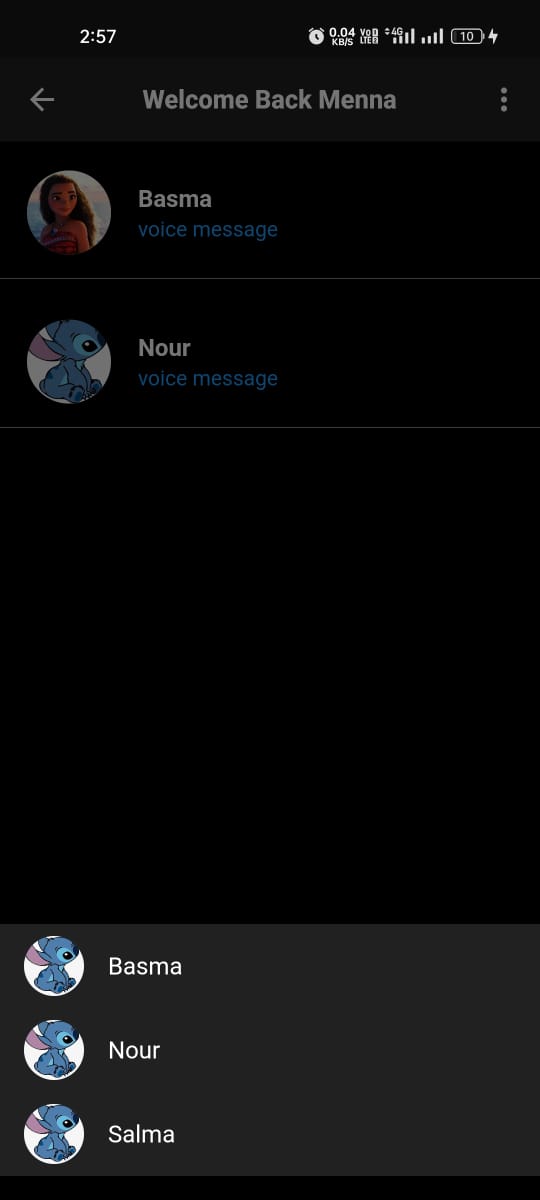
On this screen, the user can choose which chat between them and another user to enter to talk with that user. The user can also add a new user to the chat list.



**Figure ‎5.4 Chat Home Page**

**5.5 Add New User to The Chat List**

The user presses the “+” button to add someone to their chat list from their contacts.



**Figure ‎5.5 User Adding People into Chat List Using Contacts List**

**5.6 Chat**

On this screen, the user can send texts or voice notes seen in Figure 5.5 and they can also transcribe the speech or return the speech without stuttering using a Text-To-Speech API seen in Figure 5.6.

A screenshot of a phone

Description automatically generatedA black background with many small icons

Description automatically generated with medium confidence

**Figure ‎5.6 "Microphone With Checkmark" Button for Conversion of Voice Note into Text or Speech.**

**Figure ‎5.7 Microphone and Keyboard Input Options for Chat**

**5.6.1 Send Text**

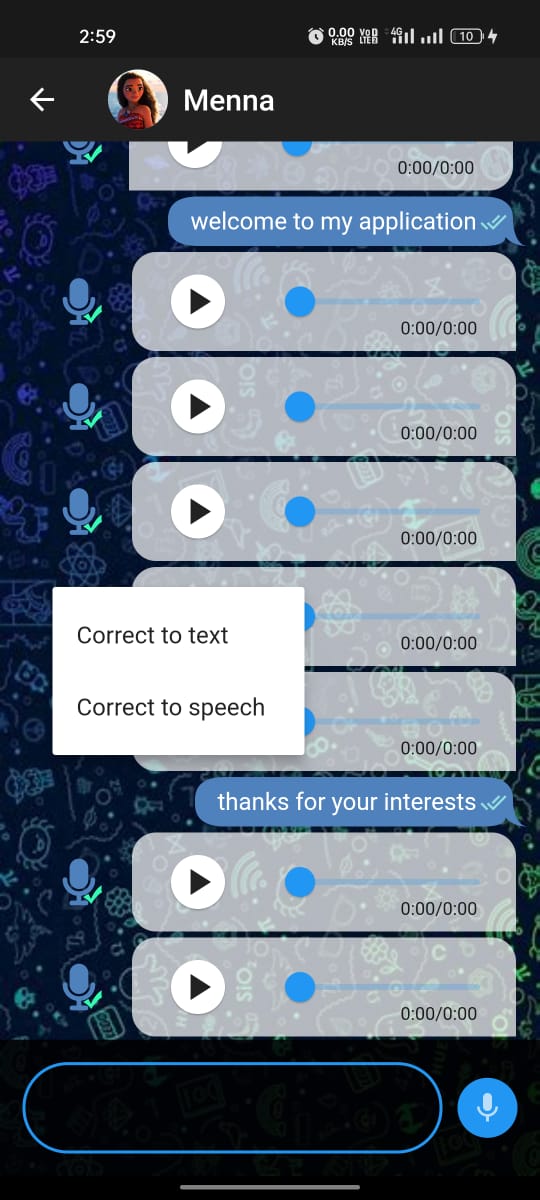
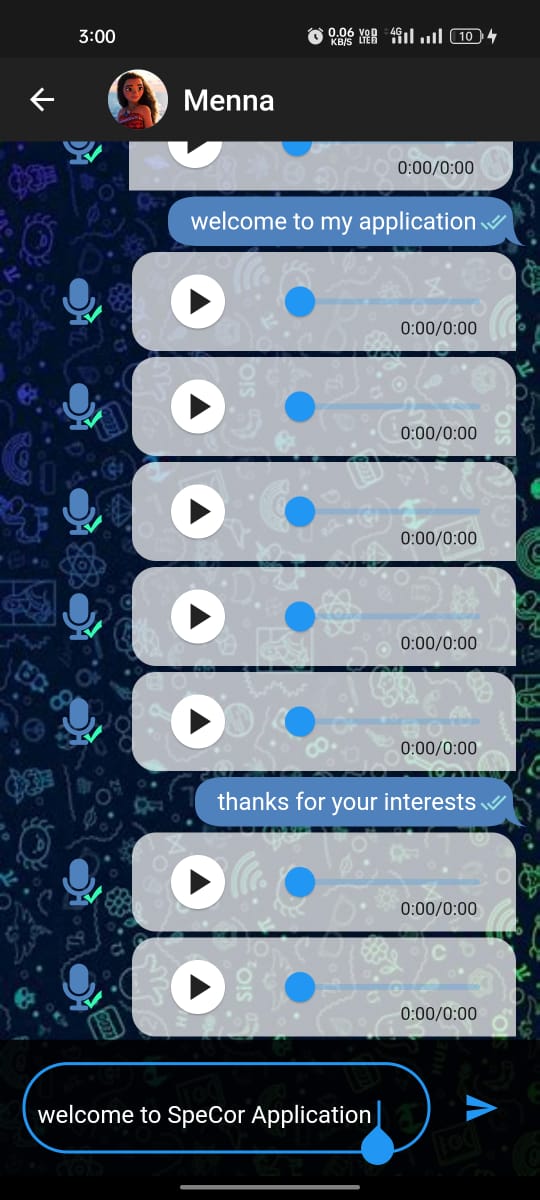
A screenshot of a cell phone

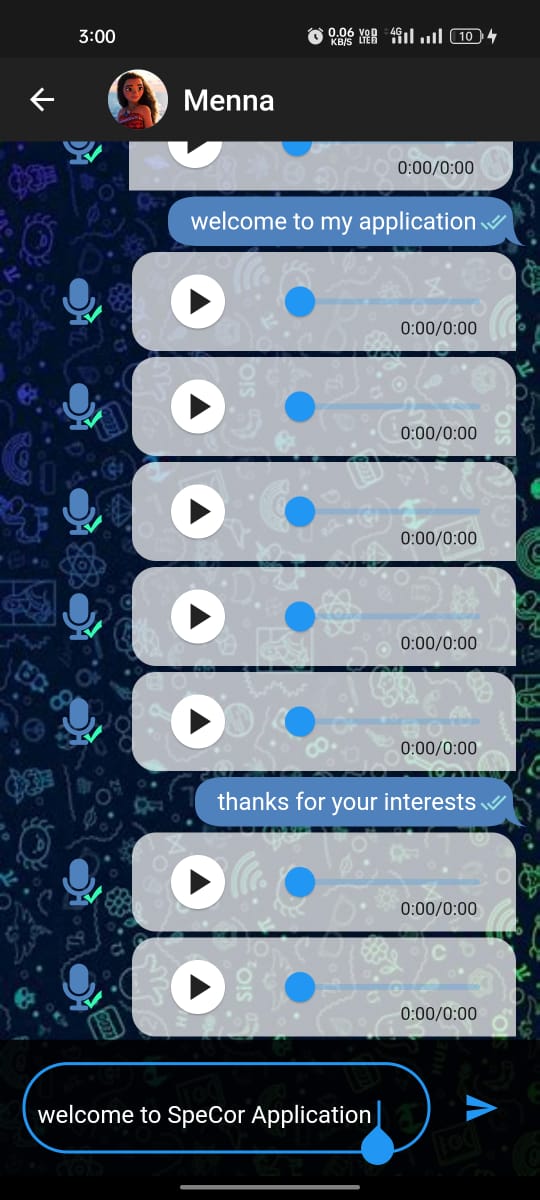
Description automatically generatedTo send a text the user has to press the oval-shaped button and type the text message and press send.

****

**Figure ‎5.8 User Sending Text Message**

**5.6.2 Return as a Text**

To return the voice note as a text, the other user must have sent the voice note then the user presses the “microphone and checkmark” icon next to the voice note to choose to return it as text.



**Figure ‎5.9 User Choosing Text Option**

**5.6.3 Return as a Speech**

To return the voice note as a speech, the other user must have sent the voice note then the user presses the icon with the microphone and checkmark to choose to return it as a speech.

A screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generated

**Figure ‎5.10 User Choosing Speech Option**

# **Chapter Six:** [**Conclusion & Future Work**](https://docs.google.com/document/d/1QXsvFDZSWKm3FPuEzFANYijyNhfdUSR3/edit#heading=h.3bj1y38)

## **Conclusion**

The exploration of Automatic Speech Recognition (ASR) technology underscores the transformative potential of advanced speech-to-text systems in enhancing communication, accessibility, and inclusivity. SpeCor addresses the critical challenges of accurately transcribing speech impediments such as stuttering and lisping, essential for making ASR technology more reliable and inclusive. Key insights reveal significant advancements in ASR due to improvements in machine learning, neural networks, and computational power. SpeCor offers an innovative solution by incorporating mechanisms to correct stuttering and lisping, integrating advanced machine learning algorithms trained on diverse datasets to recognize and correct errors associated with speech disorders and accents.

SpeCor aims to improve communication by providing accurate transcriptions, increasing accessibility for individuals with hearing impairments and other disabilities, aiding students and professionals with speech-to-text conversion, and facilitating accurate transcriptions for media and digital communication. Detailed objectives and approaches include implementing advanced signal processing techniques, training machine learning models on diverse datasets, developing real-time correction algorithms, and focusing on user-friendly design. Case studies and supporting research highlight the effectiveness of Whisper models, machine learning for stuttering identification, state-of-the-art ASR and TTS technologies, assistive technologies for accessibility, and the FastCorrect error correction model, all contributing to the development of more robust and inclusive ASR systems.

## **Future Work**

Developing a text-to-speech feature that replicates the sender’s voice can be achieved through advancing one-shot speech synthesis models. To enhance inclusivity, these models should be trained on diverse datasets that include a wide range of speech impediments. Additionally, leveraging powerful computational resources will be crucial to effectively train and optimize these models, ensuring robust performance across different speech variations and improving overall synthesis quality.

## **References**

[1] Z. Yao, L. Guo, X. Yang, W. Kang, F. Kuang, Y. Yang, Z. Jin, L. Lin, D. Povey, “Zipformer: A faster and better encoder for automatic speech recognition,” arXiv.org. https://arxiv.org/abs/2310.11230 (accessed Jun. 27, 2024).

[2] D. Prabhu, P. Jyothi, S. Ganapathy, and V. Unni, “Accented Speech Recognition With Accent-specific Codebooks,” arXiv.org. https://arxiv.org/abs/2310.15970 (accessed Oct. 26, 2023).

[3] H. O. Toyin, A. Djanibekov, A. Kulkarni, and H. Aldarmaki, “ArTST: Arabic Text and Speech Transformer,” arXiv.org. https://arxiv.org/abs/2310.16621v1 (accessed Jun. 27, 2024).

[4] C. Wang, Y. Tang, X. Ma, A. Wu, D. Okhonko, and J. M. Pino, “fairseq S2T: Fast Speech-to-Text Modeling with fairseq,” arXiv (Cornell University), Oct. 2020, doi: https://doi.org/10.48550/arxiv.2010.05171.

[5] J. Ao, R. Wang, L. Zhou, C. Wang, S. Ren, Y. Wu, S. Liu, T. Ko, Q. Li, Y. Zhang, Z. Wei, Y. Qian, J. Li, and F. Wei, “SpeechT5: Unified-Modal Encoder-Decoder Pre-Training for Spoken Language Processing,” arXiv:2110.07205 [cs, eess], May 2022, Available: https://arxiv.org/abs/2110.07205.

[6] Y. Leng, X. Tan, W. Liu, K. Song, R. Wang, X.-Y. Li, T. Qin, E. Lin, And T.-Y. Liu, “SoftCorrect: Error Correction with Soft Detection for Automatic Speech Recognition,” arXiv.org, Dec. 02, 2022. https://arxiv.org/abs/2212.01039

[7] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust Speech Recognition via Large-Scale Weak Supervision,” arXiv:2212.04356 [cs, eess], Dec. 2022, Available: https://arxiv.org/abs/2212.04356.

[8] S. Dutta, S. Jain, A. Maheshwari, G. Ramakrishnan, and Preethi Jyothi, “Error Correction in ASR using Sequence-to-Sequence Models,” arXiv (Cornell University), Feb. 2022, doi: https://doi.org/10.48550/arxiv.2202.01157.

[9] L. Meng, J. Xu, X. Tan, J. Wang, T. Qin, and B. Xu, “MixSpeech: Data Augmentation for Low-resource Automatic Speech Recognition,” arXiv:2102.12664 [cs, eess], Feb. 2021, Available: https://arxiv.org/abs/2102.12664

[10] Y. Hu, X. Jing, Y. Ko, and J. T. Rayz, “Misspelling Correction with Pre-trained Contextual Language Model,” arXiv:2101.03204 [cs], Jan. 2021, Available: https://arxiv.org/abs/2101.03204

[11] Y. Leng, X. Tan, L. Zhu, J. Xu, R. Luo, L. Liu, T. Qin, X.-Y. Li, E. Lin, and T.-Y. Liu, “FastCorrect: Fast Error Correction with Edit Alignment for Automatic Speech Recognition,” arXiv (Cornell University), May 2021, doi: https://doi.org/10.48550/arxiv.2105.03842.

[12] O. Basystiuk, N. Shakhovska, V. Bilynska, O. Syvokon, O. Shamuratov, and V. Kuchkovskiy, “The Developing of the System for Automatic Audio to Text Conversion,” ceur-ws.org. https://ceur-ws.org/Vol-2824/, Paper 1, (2018)

[13] A. Mohamed, Dmytro Okhonko, and L. Zettlemoyer, “Transformers with convolutional context for ASR,” arXiv (Cornell University), Jan. 2019, doi: https://doi.org/10.48550/arxiv.1904.11660.

[14] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, “Conformer: Convolution-augmented Transformer for Speech Recognition,” arXiv:2005.08100 [cs, eess], May 2020, Available: https://arxiv.org/abs/2005.08100

[15] M. C. Stoian, S. Bansal, and S. Goldwater, “Analyzing ASR pretraining for low-resource speech-to-text translation,” arXiv.org, Feb. 09, 2020. https://arxiv.org/abs/1910.10762

[16] J. Li, V. Lavrukhin, B. Ginsburg, R. Leary, O. Kuchaiev, J. M. Cohen, H. Nguyen, and R. T. Gadde, “Jasper: An End-to-End Convolutional Neural Acoustic Model,” arXiv:1904.03288 [cs, eess], Aug. 2019, Available: https://arxiv.org/abs/1904.03288

[17] A. Dash, N. Subramani, T. Manjunath, V. Yaragarala, and S. Tripathi, “Speech Recognition and Correction of a Stuttered Speech.” Accessed: Jun. 27, 2024. [Online]. Available: https://e-tarjome.com/storage/panel/fileuploads/2022-07-06/1657094472\_e16845.pdf

[18] Y. Miao, M. Gowayyed, and F. Metze, “EESEN: End-to-End Speech Recognition using Deep RNN Models and WFST-based Decoding,” arXiv (Cornell University), Jul. 2015, doi: https://doi.org/10.48550/arxiv.1507.08240.

[19] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, “Librispeech: An ASR corpus based on public domain audio books,” 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Apr. 2015, doi: https://doi.org/10.1109/icassp.2015.7178964.

[20] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, “AISHELL-1: An Open-Source Mandarin Speech Corpus and A Speech Recognition Baseline,” arXiv:1709.05522 [cs], Sep. 2017, Available: https://arxiv.org/abs/1709.05522

[21] T. Kourkounakis, A. Hajavi and A. Etemad, "FluentNet: End-to-End Detection of Stuttered Speech Disfluencies With Deep Learning," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 2986-2999, 2021, doi: 10.1109/TASLP.2021.3110146.

[22] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, G. Weber, “Common Voice: A Massively-Multilingual Speech Corpus,” arXiv.org, Mar. 05, 2020. https://arxiv.org/abs/1912.06670

[23] R. Cattoni, M. A. Gangi, L. Bentivogli, M. Negri, and M. Turchi, “MuST-C: A multilingual corpus for end-to-end speech translation”. In Computer Speech & Language Journal, 2020, doi: https://doi.org/10.1016/j.csl.2020.101155

[24] B. Zhang, H. Lv, P. Guo, Q. Shao, C. Yang, L. Xie, X. Xu, H. Bu, X. Chen, C. Zeng, D. Wu, Z. Peng, “WenetSpeech: A 10000+ Hours Multi-domain Mandarin Corpus for Speech Recognition,” arXiv.org, Feb. 23, 2022. https://arxiv.org/abs/2110.03370 (accessed Jun. 27, 2024).

[25] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, “wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations,” arXiv:2006.11477 [cs, eess], Oct. 2020, Available: https://arxiv.org/abs/2006.11477

‌