CONVERSION OF SPEECH INTO TEXT USING NLP

A MINI PROJECT REPORT

Submitted by

BALAJEEY RK	**(721220243005)**
HARISH KUMAR M	**(721220243017)**
THARANESH BALAJI SP	**(721220243059)**
YUVAN KUMAR P	**(7212202043064)**

In partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KARPAGAM INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY :: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this mini project report "**CONVERSION OF SPEECH INTO TEXT USING NLP**" is the bonafide work of **BALAJEEY RK (721220243005), HARISH KUMAR M (721220243017), THARANESH BALAJI SP (721220243059), YUVAN KUMAR P(721220243064)** who carried out the mini project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. R. NALLAKUMAR M.E, MBA,Ph.D.,	**Dr. R. NALLAKUMAR M.E, MBA,Ph.D.,**
SUPERVISOR	**HEAD OF THE DEPARTMENT**
Associate Professor,	Associate Professor,
Department of Artificial Intelligence and Data science,	Department of Artificial Intelligence and Data science,
Karpagam Institute of Technology,	Karpagam Institute of Technology,
Coimbatore-641105.	Coimbatore-641105.

Submitted for the university mini project Viva-voce examination conducted at Karpagam Institute of Technology, Coimbatore, on ……………

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

With genuine humility, we are obediently thankful to God Almighty without him, this work would have never been a reality.

We express our profound gratitude to our respected Chairman **Dr.R.Vasanthakumar,** for giving us this opportunity to pursue this course. At this pleasing moment of having successfully completed the mini project's work.

We wish to acknowledge sincere gratitude and heartfelt thanks to our respected Principal **Dr.P.Manimaran Ph.D.,** andVice-Principal **Dr.D.Bhanu M.E, Ph.D.,** for having given us adequate support and opportunity for completing the mini project work successfully.

We express our deep sense of gratitude and sincere thanks to our beloved Head of the Department **Dr.R.NALLAKUMAR M.E,Ph.D.,** who has been a spark for enlightening our knowledge.

Our profound gratitude goes to the mini project coordinator and mini project guide **Mr.M.VIGNESH M.E.,** and review members and all the faculty members of the Department of IT for the invaluable knowledge they have imparted to us.

Our humble gratitude and hearty thanks go to our family members and friends for their encouragement and support throughout the course of this mini project.

BALAJEEY RK (721220243005)

HARISH KUMAR M(721220243017)

THARANESH BALAJI(721220243059)

YUVAN KUMAR P(721220243064)

ABSTRACT

Create an online platform or website where customers can access the design tools. This platform should be user-friendly and interactive, allowing customers to create and visualize their abstract costume designs. Provide a wide range of design elements such as shapes, colours, patterns, and textures that customers can use to create their abstract designs. These elements should be customizable and combinable to allow for maximum creativity. Develop intuitive design tools that customers can use to manipulate and arrange the design elements. These tools can include drag-and-drop functionality, resizing, with rotating, layering, and other features to help customers create their desired abstract costume designs. Offer a library of pre-designed abstract elements and templates to inspire customers and help them get started.

This can include abstract motifs, artistic patterns, and unique textures that can be incorporated into the costume design. Provide customization options for specific costume features such as the type of fabric, silhouette, accessories, and embellishments. Once the design is finalized, customers should be able to place an order for their custom-designed costume. Provide clear pricing information, size options, and shipping details to ensure a smooth transaction process. Once an order is received, proceed with the delivery.

TABLE OF CONTENT

LIST OF ABBREVIATIONS

ASR:	AUTOMATIC SPEECH RECOGNITION
NLP -	NATURAL LANGUAGE PROCESSING
POS -	PART OF SPEECH
NER -	NAMED ENTITY RECOGNITION
LDA -	LATENT DIRICHLET ALLOCATION
NMF -	NON NEGATIVE MATRIX FACTORIZATION
GPT -	GENRECTIVE PRE TRAINED TRANSFORMER
RELU -	RECTIFIED LINEAR UNIT
GMM -	GAUSSIAN MIXTURE MODEL
DNN -	DEEP NEURAL NETWORKS
SDK	SOFTWARE DEVELOPMENT KIT
MR –	MACHINE READING
CSV -	COMMA- SEPARATED VALUES
JSON –	JAVASCRIPT OBJECT NOTATAION

LIST OF FIGURES

CHAPTER 1

1. INTRODUCTION

1.1 OVERVIEW

The problem we are addressing is the difficulties faced by vernacular students who primarily speak and understand a regional language when studying subjects taught in a different language, such as English. This language barrier creates a significant obstacle for these students, impeding their ability to comprehend and engage with the material being presented. As a result, they often struggle to perform well academically and may experience a lack of confidence in their learning abilities. To tackle this issue, our proposed solution involves leveraging Natural Language Processing (NLP) techniques to convert speech to text. By implementing a speech-to-text conversion system, we aim to provide vernacular students with the means to interact with educational content in their native language, thereby bridging the language gap and enhancing their learning experience.

Through speech-to-text conversion using NLP, vernacular students gain several benefits. Firstly, they can listen to lectures or educational content in a language they are comfortable with and receive written transcripts, allowing for improved comprehension and understanding. This empowers them to engage actively in the learning process and promotes a more inclusive educational environment. Moreover, the converted text can be utilized to create study materials, notes, and digital resources in vernacular languages. This enhances accessibility and ensures that vernacular students have access to learning materials tailored to their language needs. Additionally, the converted text can be further processed using NLP techniques to facilitate personalized learning experiences.

This includes generating summaries, creating practice questions, or recommending supplemental resources tailored to each student's unique requirements. By utilizing NLP and speech-to-text conversion, our solution strives to overcome the language barrier faced by vernacular students, providing them with equal opportunities to succeed in their education and fostering a more inclusive and supportive learning environment.

1.2 EXISTING SYSTEM

The existing system for vernacular students facing difficulties in studying subjects taught in a different language typically lacks a dedicated solution. In traditional educational settings, these students often have limited resources and support to overcome the language barrier. In many cases, vernacular students are expected to adapt to the language of instruction without adequate assistance. They may struggle to understand lectures, textbooks, and other educational materials presented in a foreign language.

This can lead to difficulties in comprehension, reduced engagement, and lower academic performance. Existing approaches to address this issue often rely on manual translation efforts or human interpretation, which can be time-consuming, costly, and prone to errors. Teachers or peers may provide ad hoc translation or interpretation support, but this approach is not scalable and may not be readily available in all learning environments. Some educational institutions may offer language learning programs or bilingual education models, but these often require substantial time and effort to achieve proficiency in the second language, which may not be feasible for all students.

1.3 PROPOSED SYSTEM

The proposed system is a speech-to-text conversion system that utilizes Natural Language Processing (NLP) techniques to address the language barrier faced by vernacular students in studying subjects taught in a different language. The system aims to convert spoken language into written text, allowing students to interact with educational content in their native language and improving their comprehension and engagement. The system takes audio input, either through live recordings or pre-recorded files, and applies audio preprocessing techniques to enhance the quality and remove any noise or distortion.

It then utilizes Automatic Speech Recognition (ASR) algorithms to convert the audio into text. ASR algorithms, often based on deep learning models like recurrent neural networks or transformer-based models, analyze the audio input and transcribe it into written text. To enhance the accuracy and coherence of the transcriptions, the system incorporates a language model. This model considers the context and grammar of the spoken language, resulting in more accurate and meaningful transcriptions. The transcribed text is then subjected to post-processing, which involves cleaning and formatting to improve readability and correctness. Punctuation correction, capitalization, and the removal of unnecessary noise or filler words may be applied in this step.

CHAPTER 2

LITERATURE REVIEW

The literature related to speech-to-text conversion and its application for vernacular students primarily revolves around the fields of Natural Language Processing (NLP), Automatic Speech Recognition (ASR), and educational technology. Here are some key themes and research areas:

1 **Automatic Speech Recognition (ASR) Techniques:** Numerous studies focus on improving the accuracy and robustness of ASR models for different languages and dialects. Researchers explore various deep learning architectures, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformer-based models, to enhance the performance of ASR systems.

2 **Language Modeling for Vernacular Languages:** Language models tailored to specific vernacular languages play a vital role in speech-to-text conversion. Researchers investigate techniques to develop language models that capture the grammar, syntax, and context of vernacular languages, resulting in more accurate transcriptions.

3 **Speech Data Collection and Corpus Building:** Building speech corpora specifically for vernacular languages is crucial for training and evaluating ASR systems. Researchers explore methodologies for collecting and curating speech data, including crowd-sourcing techniques, to create comprehensive datasets representative of the target vernacular languages.

4 **Evaluation and Performance Metrics:** Evaluating the performance of speech-to-text conversion systems is a key focus. Researchers propose metrics to assess the accuracy, word error rate (WER), and other aspects of ASR models when applied to vernacular languages. Comparative studies may also analyze the performance of different ASR techniques and models.

5 **Educational Applications and Impact:** The literature also highlights the impact of speech-to-text conversion on vernacular students' education. Studies explore the effectiveness of using speech-to-text technology in improving comprehension, engagement, and academic performance. They examine its application in transcription of lectures, creation of study materials, and integration with other educational tools.

CHAPTER 3

3. SYSTEM SPECIFICATION

Minimum Specification

- 4 GB RAM (Minimum)
- 80 GB HDD
- Dual Core processor

Recommended Specification

- 8 GB RAM
- 120 GB SSD
- Intel Core i5-8259U, or AMD Ryzen 5 2700X (Processor)
- NVIDIA GT 1050 or Quadro P1000 (Graphic Card)

3.1 SOFTWARE SPECIFICATION

OPERATING SYSTEM: WINDOWS 10 AND ABOVE

LANGUAGES	: Python
SOFTWARE	: Pycharm, Jupyter Notebook, Spyder
SERVER	: Local Database (If requires)

PYTHON

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high-level dynamic data types, and classes. It supports multiple programming paradigms beyond object-oriented programming, such as procedural and functional programming. It uses a simplified syntax with an emphasis on natural language, for a much easier learning curve for beginners.

PYCHARM

It allows viewing of the source code in a click. Software development is much faster using PyCharm. The feature of error spotlighting in the code further enhances the development process. The community of Python Developers is extremely large so that we can resolve our queries/doubts easily. PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. It makes Python development accessible to those who are new to the world of software programming. PyCharm Community Edition is excellent for developers who wish to get more experience with Python.

JUPYTER NOTEBOOK

Jupyter Notebook allows users to compile all aspects of a data project in one place making it easier to show the entire process of a project to your intended audience. Through the web-based application, users can create data visualizations and other components of a project to share with others via the platform. The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience. Jupyter Notebook allows users to convert the notebooks into other formats such as HTML and PDF. It also uses online tools and nbviewer which allows you to render a publicly available notebook in the browser directly. Jupyter is another best IDE for Python Programming that offers an easy-to-use, interactive data science environment across many programming languages besides Python.

SPYDER

Spyder is a free and open source scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package. pyder is an open- source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open-source software.[4][5] It is released under the MIT. license.[6]Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community. Spyder is extensible with first-party and third-party plugins,[7] includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint[8] and Rope. It is available cross-platform through Anaconda, on Windows, on macOS through MacPorts, and on major Linux distributions such as Arch Linux, Debian, Fedora, Gentoo Linux, openSUSE and Ubuntu.[9][10]

LOCAL DATABASE

Local databases reside on your local drive or on a local area network. They often have proprietary APIs for accessing the data. When they are shared by several users, they use file-based locking mechanisms. Because of this, they are sometimes called file-based databases. The Oracle. Oracle is the most widely used commercial relational database management system, built-in assembly languages such as C, C++, and Java. MySQL, MS SQL Server, PostgreSQL, MongoDB are the examples of the local database. Personal database system is the local database system which is only for one user to store and manage the data and information on their own personal system. There are number of applications are used in local computer to design and managed personal database system.

CHAPTER 4

DESIGN METHODOLOGY

4. SYSTEM DESIGN:

Designing a system for speech-to-text conversion to address the language barrier for vernacular students involves multiple components and considerations. Here's a high-level overview of the system design:

Input Source: The system should support different input sources, such as live recordings from microphones or pre-recorded audio files. It should provide mechanisms to capture and process the audio input.

Audio Preprocessing: Apply audio preprocessing techniques to enhance the quality of the input audio and remove any noise or distortion. Common preprocessing steps may include noise reduction, normalization, and filtering.

Speech Recognition: Utilize Automatic Speech Recognition (ASR) techniques to convert the audio into text. ASR models based on deep learning algorithms, such as recurrent neural networks (RNNs) or transformer-based models, can be employed for accurate transcription.

Language Model: Enhance the ASR output by incorporating a language model. The language model considers the context and grammar of the vernacular language spoken by the students, improving the accuracy and coherence of the transcriptions.

Post-processing and Formatting: Clean and format the transcribed text to improve readability and correctness. This step may involve tasks like punctuation correction, capitalization, and removal of unnecessary noise or filler words.

Storage and Retrieval: Store the transcriptions in a suitable format and structure for future retrieval and usage. This may involve storing the text in a database or file system, associating it with metadata such as timestamps, and providing mechanisms for efficient retrieval.

4.1 PROBLEM DEFINITION

The problem is that vernacular students face difficulties studying subjects taught in a language different from their native language. This language barrier hinders their comprehension, engagement, and overall academic performance. The existing educational system often lacks dedicated solutions to address this issue, resulting in limited resources and support for these students to overcome language-related challenges.

The existing methodology for addressing the language barrier faced by vernacular students involves manual translation efforts, limited language support, and ad hoc assistance. Some common approaches include:

Manual Translation: Teachers or peers may provide ad hoc translation or interpretation support during lectures or when studying materials. However, this approach is time-consuming, lacks scalability, and may not be readily available in all learning environments.

Bilingual Education: Some educational institutions offer bilingual education programs where students receive instruction in both their native language and the language of instruction. However, these programs require significant time and effort to achieve proficiency in the second language, making them impractical for all students.

Language Learning Programs: Students may be enrolled in language learning programs to improve their proficiency in the language of instruction. While these programs can be beneficial, they require dedicated time and resources, which may not be feasible for all students.

Educational Support: Teachers may provide additional support, such as simplified explanations, visual aids, or supplementary materials, to help vernacular students overcome language barriers. However, this support may not be standardized across all educational settings, leading to inconsistent assistance.

4.2 PROPOSED METHODOLOGY:

The proposed methodology aims to address the language barrier faced by vernacular students through the use of speech-to-text conversion using Natural Language Processing (NLP) techniques. The following is a brief description of the proposed methodology. The first step is to clearly identify the challenges faced by vernacular students due to the language barrier and understand their specific needs and requirements. This helps in designing a targeted solution that caters to their unique circumstances. Next, a diverse and representative dataset of speech recordings in the vernacular language(s) spoken by the students is collected. This dataset serves as the foundation for training an Automatic Speech Recognition (ASR) model that can accurately convert spoken language into text.

Requirement Analysis: Begin by understanding the specific requirements and challenges faced by vernacular students in their learning environment. Identify the target vernacular language(s), the context in which the system will be used, and the desired outcomes.

Data Collection: Collect a diverse and representative dataset of audio recordings in the vernacular language(s) spoken by the students. Ensure the dataset covers a wide range of accents, dialects, and speaking styles to train a robust speech recognition model.

Preprocessing: Apply preprocessing techniques to the audio data to enhance its quality. This may involve denoising, normalization, and segmenting the recordings into manageable units.

Speech Recognition Model: Train or select an appropriate Automatic Speech Recognition (ASR) model for the vernacular language(s). This may involve exploring different architectures, such as deep neural networks, and adapting them to the specific language characteristics.

Language Modeling: Develop language models tailored to the vernacular language(s) to improve the accuracy and coherence of the transcriptions. Incorporate linguistic rules, grammar, and contextual information into the language model to generate more meaningful and natural transcriptions.

Performance Optimization: Optimize the system's performance by considering factors such as computational efficiency, scalability, and real-time processing capabilities. Explore techniques like parallelization, distributed computing, or cloud-based solutions to ensure efficient and responsive operation.

OVERALL ARCHITECTURE OF SYSTEM

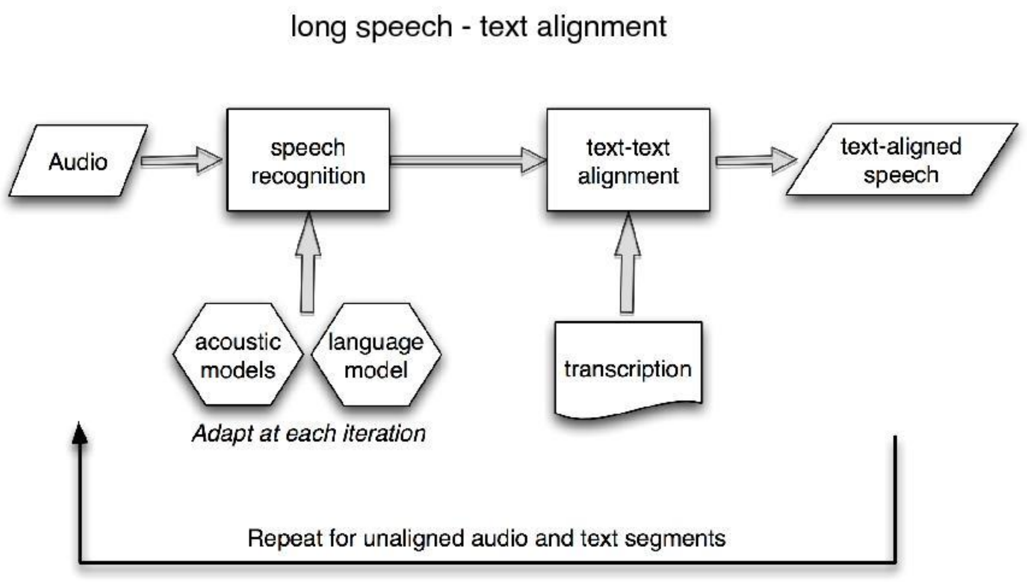


Fig 4.1 Overall Architecture of System

The architecture diagram for the proposed speech-to-text conversion project consists of several interconnected modules. At the center of the architecture is the Automatic Speech Recognition (ASR) module, which transcribes the audio input into text using a trained ASR model. The input module captures the audio input, which undergoes preprocessing to enhance its quality. The language modeling module incorporates language models specific to the vernacular language(s) to improve transcription accuracy.

The post-processing module refines and formats the transcriptions, making them cleaner and more readable. The storage and retrieval module handles the storage and management of the transcriptions, and the user interface module provides an intuitive interface for user interaction. Performance optimization techniques are applied to ensure efficient and real-time processing. Overall, the architecture follows a modular design, enabling accurate and accessible speech-to-text conversion for vernacular students.

4.3 DATA PRE-PROCESS:

Data preprocessing plays a crucial role in the speech-to-text conversion project. It involves preparing the audio data for effective processing and improving the quality of the input. The following steps are commonly involved in data preprocessing for this project:

Noise Removal: Audio data often contains background noise that can negatively impact the accuracy of speech recognition. Noise removal techniques, such as spectral subtraction or wavelet denoising, can be applied to reduce noise and enhance the quality of the speech signal.

Normalization: Normalizing the audio data helps to bring it to a standard level of amplitude. This ensures that the speech recognition model performs consistently across different recordings. Techniques like peak normalization or RMS normalization can be employed for this purpose.

Segmentation: Long audio recordings are typically segmented into smaller units for efficient processing. Segmentation can be based on silence detection or fixed time intervals. This helps in better alignment of the transcriptions with the original speech segments and facilitates parallel processing.

Resampling: In some cases, it may be necessary to adjust the sampling rate of the audio data to match the requirements of the speech recognition model. Resampling techniques, such as interpolation or decimation, can be used to modify the sampling rate while preserving the essential characteristics of the speech signal.

Feature Extraction: Extracting relevant features from the audio data is a critical step in speech recognition. Commonly used features include Mel Frequency Cepstral Coefficients (MFCCs) and their derivatives. These features capture the spectral characteristics of the speech signal and are fed into the speech recognition model.

Data Augmentation: Data augmentation techniques can be applied to increase the diversity and size of the training dataset. This can involve techniques like adding background noise, varying pitch or speed, or applying simulated room acoustics to the audio data.

4.4 NATURAL LANGUAGE PROCESSING:

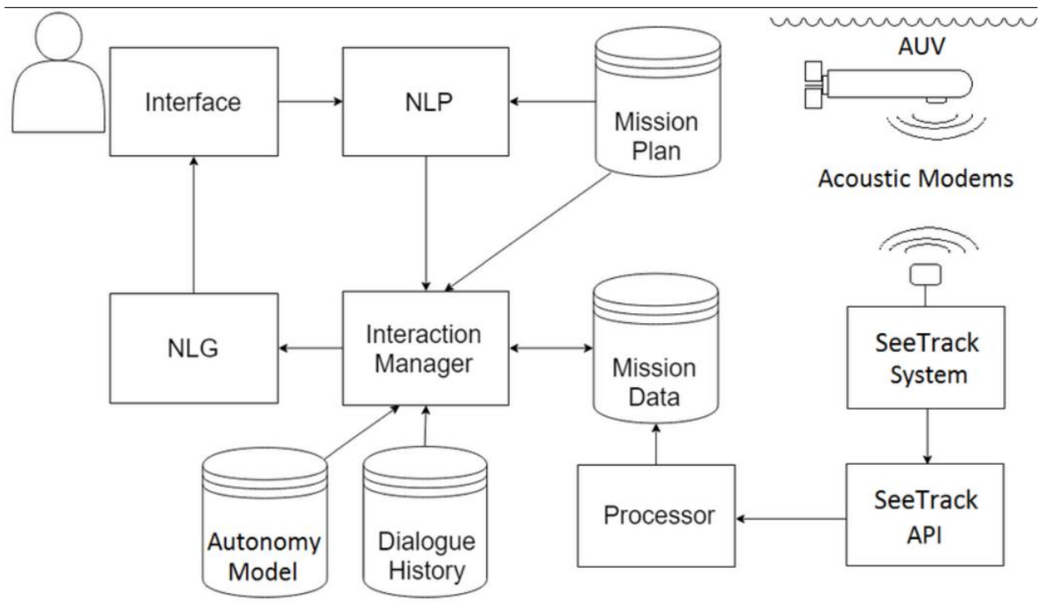


Fig 4.2 NLP Architecture

Within the NLP (Natural Language Processing) component of the speech-to-text conversion project, several modules are involved to enhance the accuracy and understanding of the transcribed text. The key modules within NLP include:

4.4.1 Tokenization:

Tokenization is the process of breaking down a text into individual tokens, which can be words, subword units, or characters. It forms the fundamental step in NLP as it allows further processing and analysis of the text. Tokenization is typically performed using whitespace or punctuation as delimiters, but more advanced techniques like subword tokenization (e.g., Byte-Pair Encoding or WordPiece) can handle morphologically rich languages or unknown words effectively.

4.4.2 Part-of-Speech (POS) Tagging:

POS tagging is the task of assigning grammatical labels, or tags, to each token in a sentence, indicating its syntactic category (e.g., noun, verb, adjective, etc.). POS tagging provides essential information about the role of each word in the sentence and aids in understanding the grammatical structure of the transcribed text. This step is typically achieved using statistical models or neural networks trained on annotated corpora.

4.4.3 Named Entity Recognition (NER):

NER identifies and classifies named entities in the text, such as person names, locations, organizations, dates, and other proper nouns. NER is crucial for information extraction and understanding the context of the transcribed text. It involves training models using machine learning algorithms, such as conditional random fields or deep learning models, on labeled datasets.

4.4.4 Dependency Parsing:

Dependency parsing aims to analyze the grammatical structure of a sentence by determining the dependencies between words. It assigns syntactic relationships, such as subject, object, modifier, etc., to each word, creating a parse tree representation. Dependency parsing provides insights into the syntactic structure of the transcribed text and is often accomplished using graph-based or transition-based parsing algorithms.

4.4.5 Sentiment Analysis:

Sentiment analysis focuses on determining the sentiment or opinion expressed in a piece of text. It classifies the sentiment as positive, negative, or neutral, providing insights into the emotional tone of the transcribed text. Sentiment analysis can be approached using supervised learning algorithms, lexicon-based methods, or deep learning models trained on sentiment-labeled data.

4.4.6 Language Modeling:

Language modeling is the task of predicting the next word in a sequence of words based on the context. It models the probability distribution of words in a given context, capturing the linguistic rules and patterns of the language. Language models, such as n-gram models or transformer models, are trained on large text corpora and are used to generate more coherent and contextually appropriate transcriptions.

4.4.7 Topic Modeling:

Topic modeling aims to discover the underlying topics or themes within a collection of documents. It is useful for organizing and summarizing textual data.Techniques for topic modeling include Latent Dirichlet Allocation (LDA) and Non-negative Matrix Factorization (NMF), which identify latent topics based on word co-occurrence patterns.

4.4.8 Text Summarization:

Text summarization involves condensing a long piece of text into a shorter, concise summary while preserving its main ideas and key information. Extractive summarization selects and combines important sentences or phrases from the text, while abstractive summarization generates new sentences that capture the essence of the original text. Summarization techniques can be based on statistical algorithms or neural network models trained on large datasets.

4.4.9 Machine Translation:

Machine translation aims to automatically translate text from one language to another. It involves training models on parallel corpora, which are collections of texts in multiple languages, to learn the mappings between words and phrases in different languages. Machine translation can be performed using rule-based approaches, statistical models (such as phrase-based models), or neural network models (such as sequence-to-sequence models with attention mechanisms).

4.4.10 Text Generation:

Text generation focuses on generating coherent and contextually appropriate text based on a given prompt or input. It can be used for chatbots, dialogue systems, or creative writing applications. Text generation techniques include language models, such as the GPT (Generative Pre-trained Transformer) model, which are trained on large corpora and can generate text based on the learned patterns and context.

4.5 STORAGE RETRIEVAL AND DATA MODELING:

The Storage and Retrieval module in the speech-to-text conversion project is responsible for managing the storage, organization, and retrieval of the transcribed text data. It ensures efficient storage and quick access to the converted text, allowing users to retrieve and utilize the transcriptions as needed. encoding. The encoder typically consists of multiple hidden layers that gradually reduce the dimensionality of the input data. Each hidden layer applies linear transformations and non- linear activation functions to capture relevant features and compress the data. The final layer of the encoder produces the latent representation, which contains a compressed and encoded version of the input data.

The module performs the following functions:

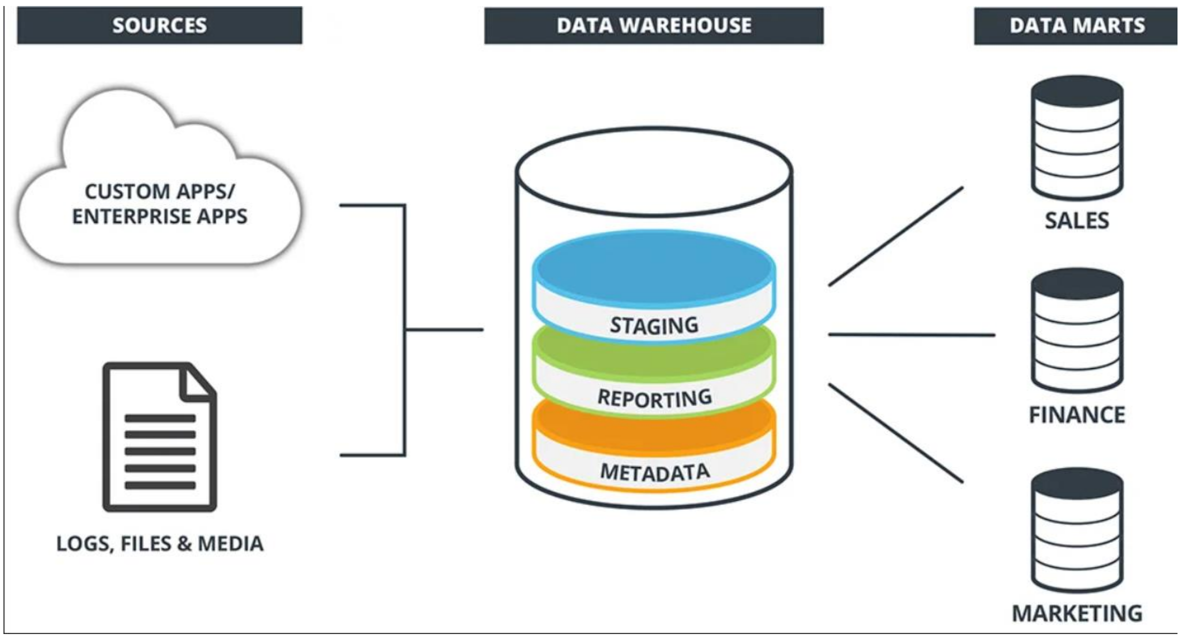


Fig 4.3 Storage Retrieval System

Storage Management: The module handles the storage of the transcribed text data in a structured manner. It may utilize databases, file systems, or cloud storage services to store the text data securely and efficiently. The storage management component ensures that the transcriptions are stored in a reliable and scalable manner, accommodating a potentially large volume of data.

Indexing: To enable fast and accurate retrieval of the transcribed text, the module employs indexing techniques. Indexing involves creating an index or catalog of the stored text data, which allows for efficient searching based on specific criteria. The index may include metadata such as timestamps, speaker identification, or keywords associated with the transcribed content.

Search and Retrieval: The module provides search and retrieval capabilities for users to locate specific transcriptions. Users can search for transcriptions based on keywords, timestamps, speaker names, or other relevant criteria. The module utilizes the indexing information to quickly retrieve and present the desired transcriptions to the user.

Sorting and Filtering: The module may offer sorting and filtering options to facilitate easy access to the transcribed text. Users can sort the transcriptions based on different parameters such as date, relevance, or speaker. Filtering options allow users to narrow down the search results based on specific criteria, such as specific speakers or topics.

Data Backup and Recovery: To ensure data integrity and prevent loss, the module incorporates backup and recovery mechanisms. Regular backups of the transcribed text data are performed, and data recovery options are available in case of any system failures or data corruption incidents. This ensures the preservation of the transcriptions and minimizes the risk of data loss.

Loss Function: A loss function is essential for training an autoencoder. It quantifies the difference between the input data and the reconstructed output generated by the decoder. The choice of the loss function depends on the nature of the input data. For example, mean squared error (MSE) is commonly used for continuous data, while binary cross-entropy is suitable for binary data. The loss function guides the training process by minimizing the reconstruction error, driving the autoencoder to learn a compact representation that accurately reconstructs the input data.

Activation Functions: Activation functions play a vital role in the hidden layers of the encoder and decoder. They introduce non-linearity to the autoencoder, allowing it to capture complex patterns and relationships in the data. Commonly used activation functions include the rectified linear unit (ReLU), sigmoid, and hyperbolic tangent (tanh). These functions help in modeling the non-linear mapping between the input and the latent space, as well as the non-linear reconstruction process from the latent space to the out.

4.6 EXPORTING THE MODEL

The final step of the speech-to-text conversion project involves presenting the transcribed text to the user in a usable and accessible format. This step focuses on delivering the converted text output in a convenient and meaningful way to facilitate its utilization by the intended audience.

Text Formatting: The transcribed text may undergo formatting to improve its visual presentation and readability. This can involve adjusting font size, spacing, or alignment, and applying appropriate formatting styles such as bold or italics for emphasis.

Segmentation and Organization: If the transcribed text is long or contains multiple sections, it may be segmented and organized into smaller, manageable units. This could include dividing it into paragraphs, chapters, or subsections, depending on the structure of the original speech content.

Timestamps and Synchronization: If the original audio had timestamps indicating the temporal alignment of the transcribed words, these timestamps can be utilized to synchronize the text with the corresponding audio. This allows users to navigate through the text and refer back to the original audio content when needed.

Presentation Options: The final step may involve providing users with options for customizing the presentation of the transcribed text. Users may be able to choose font styles, color schemes, or line spacing according to their preferences. Additionally, they may have the option to select different viewing modes, such as a full-screen mode or a compact mode for mobile devices.

Export and Sharing: To facilitate further use and distribution of the transcribed text, the final step may include options for exporting or sharing the output. Users may have the ability to download the text in different file formats (e.g., PDF, TXT) or share it directly through email, messaging apps, or social media platforms.

Accessibility Features: Consideration for accessibility is important in the final step. The transcribed text should be made accessible to users with disabilities. This may involve providing features such as screen reader compatibility, adjustable text size and color contrast, and support for alternative input methods.

CHAPTER 5

IMPLEMENTATION

5.1 MODULE DESCRIPTION

The NLP processing module is the core of the project, where the actual speech-to-text conversion takes place. It includes various sub-modules and techniques such as automatic speech recognition (ASR) for converting the audio input into text, language modeling to enhance transcription accuracy, and linguistic analysis for interpreting the structure and meaning of the transcribed text. These modules employ algorithms and models from the field of NLP to handle tasks like acoustic modeling, word segmentation, part-of-speech tagging, syntactic parsing, and semantic analysis.

5.2 INPUT MODULE:

The input of the speech-to-text conversion project is primarily audio data in the form of speech. Users provide spoken language input, typically in the form of recorded audio files or live audio streams. This audio data serves as the source material for the speech recognition and transcription process. The audio input can come from various sources, such as recorded lectures, interviews, conversations, or any other spoken content that needs to be converted into text. It may be in different formats, such as WAV, MP3, or other common audio file formats. Before being processed, the audio input undergoes a pre-processing stage. This stage involves techniques such as noise removal, normalization, and segmentation to enhance the quality of the audio signal and prepare it for accurate speech recognition. Noise removal algorithms filter out background noise and disturbances, while normalization techniques adjust the audio levels for consistency. Segmentation involves breaking the audio into smaller segments for efficient processing. Once pre-processed, the audio data is fed into the speech recognition system.

The speech recognition module utilizes various algorithms, models, and techniques to convert the spoken language into text. It analyzes the acoustic features of the speech signal, matches them with linguistic patterns, and generates a sequence of text that represents the transcribed speech. The transcribed text output may be in the form of individual words, sentences, or paragraphs, depending on the specific requirements of the project and the user's preferences. The text may also be accompanied by additional information, such as timestamps indicating the temporal alignment of the transcribed words with the original audio. In addition to the audio input, the system may also receive optional metadata or context information that can aid in the transcription process. This metadata could include information about the language being spoken, the speaker's identity, or any other relevant details that can help improve the accuracy and understanding of the transcribed text.The input of the project can be obtained through various means, such as uploading pre-recorded audio files, capturing live audio streams through microphones, or integrating with others.

Overall, the input of the speech-to-text conversion project consists of audio data, which is pre-processed and then processed through the speech recognition module to generate accurate and coherent transcriptions in the form of text.

5.3 PRE PROCESSING:

The speech-to-text conversion project is the audio preprocessing stage. After the audio input is obtained, it undergoes various pre-processing techniques to enhance the quality of the audio signal and prepare it for accurate speech recognition.

1 **Noise Removal:** Background noise, such as static, hums, or environmental sounds, can interfere with the clarity of the speech signal. Noise removal techniques, such as spectral subtraction or wavelet denoising, are applied to reduce or eliminate unwanted noise, improving the overall signal-to-noise ratio.

2 **Normalization:** Audio normalization techniques adjust the volume levels of the audio signal to ensure consistency and optimal levels for further processing. This helps to address variations in the loudness or amplitude of the audio, making it easier to analyze and recognize the speech accurately.

3 **Segmentation:** Segmentation involves breaking down the audio into smaller, manageable segments for efficient processing. The audio may be divided into segments based on silence detection, pauses, or other criteria. Segmenting the audio helps in isolating individual speech utterances, making it easier to process and transcribe them separately.

4 **Filtering:** Filtering techniques, such as high-pass or low-pass filters, may be applied to further refine the audio signal. These filters help remove unwanted frequency components that are not relevant to the speech, ensuring a cleaner and more focused input for the speech recognition system.

5 **Resampling:** Resampling is performed when the audio has a non-standard sample rate or needs to be converted to a different sample rate for compatibility with the speech recognition algorithms. This process involves adjusting the rate at which the audio signal is sampled to match the desired sample rate.

5.4 AUTOMATIC SPEECH RECOGNITION MODULE:

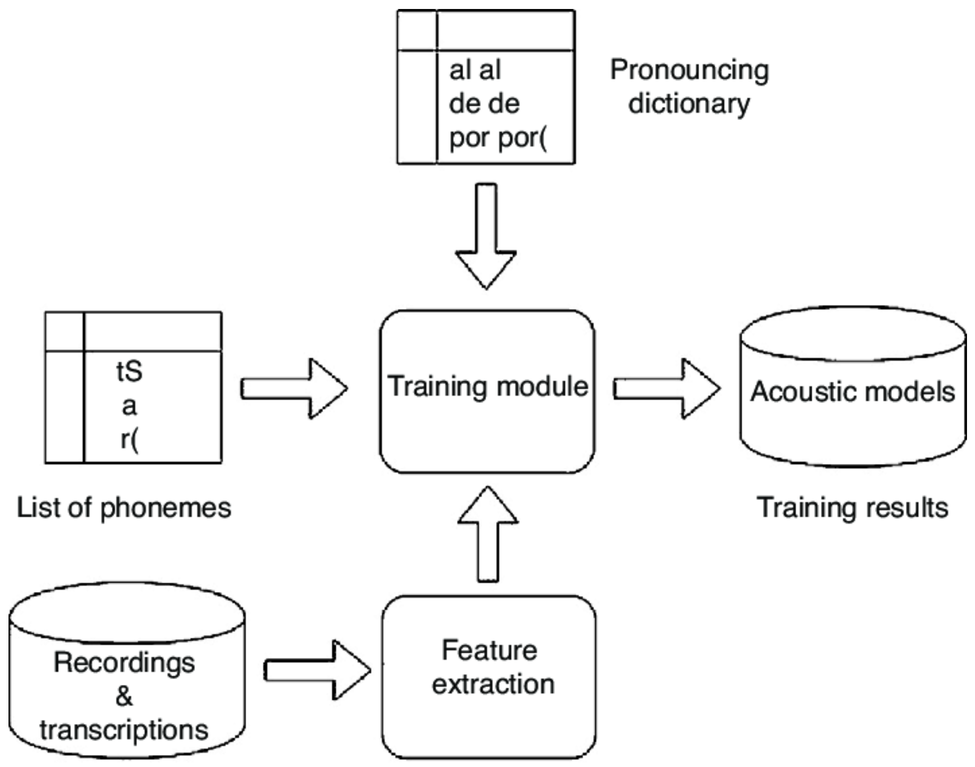


Fig 5.1 Automatic Speech Recognition Architecture

The Automatic Speech Recognition (ASR) module is a crucial component of the speech-to-text conversion project. It is responsible for converting spoken language input into written text. The ASR module utilizes advanced algorithms and models to analyze and interpret the acoustic features of the speech signal. The process of automatic speech recognition can be divided into several steps. Firstly, the audio input is pre-processed to enhance its quality and make it suitable for analysis. This may involve noise removal, normalization, and segmentation techniques to improve the signal-to-noise ratio and isolate individual speech segments. Next, the pre-processed audio is transformed into a feature representation that captures relevant acoustic information. Commonly used features include Mel-frequency cepstral coefficients (MFCCs) and filterbank energies. These features represent the spectral characteristics of the speech signal and provide a compact representation for further processing.

The ASR module utilizes a language model that contains statistical information about the language being spoken. This model helps guide the recognition process by providing information about word sequences, grammar, and word probabilities. Language models can be based on n-grams, recurrent neural networks (RNNs), or transformer models, depending on the complexity and size of the vocabulary. The acoustic features from the audio input are then compared to the acoustic models, which are trained using large amounts of labeled speech data. Acoustic models can be based on hidden Markov models (HMMs), Gaussian mixture models (GMMs), or deep neural networks (DNNs). These models learn the relationship between the acoustic features and the corresponding phonetic units, such as phonemes or subword units. During the recognition process, the ASR module generates a set of possible word hypotheses based on the acoustic and language models. These hypotheses are then scored and ranked based on their likelihoods using decoding algorithms such as the Viterbi algorithm or beam search. The most probable word sequence is selected as the final transcription. The performance of the ASR module heavily relies on the quality and diversity of the training data, the accuracy of the acoustic and language models, and the effectiveness of the decoding algorithms. Continuous research and advancements in deep learning, such as recurrent neural networks (RNNs) and transformer models, have significantly improved the accuracy and robustness of ASR systems.

Overall, the ASR module is a key component that enables the speech-to-text conversion project to accurately recognize and transcribe speech. It combines acoustic modeling, language modeling, and decoding techniques to convert spoken language input into written text, providing a valuable tool for vernacular students to access and engage with educational content.

5.5 NLP MODULE:

The first step in the NLP process is tokenization. Tokenization involves breaking down the transcribed text into individual words, sentences, or subword units. This helps in further analysis and processing of the text at a granular level. The next step is part-of-speech (POS) tagging. POS tagging assigns grammatical labels to each word in the text, such as noun, verb, adjective, etc. This information helps in understanding the syntactic structure of the text and disambiguating word meanings. Named Entity Recognition (NER) is another important NLP technique used in this project. NER identifies and classifies named entities, such as person names, locations, organizations, etc., within the text. This information can be useful for extracting relevant information and understanding the context of the transcribed content. Dependency parsing is employed to analyze the grammatical relationships between words in a sentence. It identifies the syntactic dependencies and hierarchical structure of the text, enabling a deeper understanding of the relationships between words and their roles within the sentence.

Sentiment analysis is another NLP technique that can be applied to the transcribed text. It helps in determining the emotional tone, sentiment, or polarity of the text, whether it is positive, negative, or neutral. This information can provide insights into the subjective aspects of the speech content.

Language modeling is an essential component of NLP in this project. Language models are trained on large text corpora and learn the patterns, grammar, and contextual information of the language. By applying language modeling techniques, the system can generate more coherent and contextually appropriate transcriptions. They help in capturing the syntactic, semantic, and contextual information, enabling a more comprehensive analysis of the transcribed content. By leveraging NLP, the system can go beyond basic speech recognition and provide a more in-depth and meaningful representation of the transcribed speech. This facilitates better comprehension and utilization of the transcriptions by vernacular students, aiding in their studies and improving their overall learning experience.

Post-Processing Module: The post-processing module in the speech-to-text conversion project is responsible for refining and improving the transcribed text generated by the Automatic Speech Recognition (ASR) module. While the ASR module performs the initial conversion of speech into text, the post-processing module focuses onand readability of the transcriptions.

Error Correction: The module analyzes the transcribed text and identifies errors in spelling, grammar, punctuation, or word choice. It applies spelling correction algorithms, grammar rules, and contextual information to correct these errors and make the text more accurate and linguistically correct.

Disfluency Removal: Disfluencies, such as repetitions, hesitations, or false starts, are common in spontaneous speech. The post-processing module detects and removes these disfluencies, resulting in a smoother and more coherent transcription.

Formatting and Structure: The module may apply formatting and structural adjustments to the transcribed text, such as capitalization, paragraph breaks, or sentence segmentation. This improves the readability and organization of the text, making it easier for users to comprehend and navigate.

Contextual Enhancement: The module utilizes contextual information, such as language models, semantic analysis, or knowledge bases, to improve the understanding and contextuality of the transcribed text. It resolves ambiguous words or phrases, disambiguates homophones, and fills in missing information based on the surrounding context.

Post-editing Tools: The post-processing module may provide post-editing tools or interfaces to allow users to manually review and edit the transcriptions. This enables users to correct any remaining errors, add specific domain knowledge, or make stylistic according to their requirements.

CHAPTER 6

6.1 CONCLUSION

In conclusion, the speech-to-text conversion project aims to address the challenges faced by vernacular students in studying by providing a solution that converts speech into text using NLP techniques. The project's proposed methodology includes various modules such as speech recognition, language modeling, and post-processing to ensure accurate and accessible text output. The architecture of the system follows a modular design, allowing for efficient and effective conversion of speech into text. By converting speech into text, vernacular students can overcome the difficulties they face in studying. They can access the transcribed text, review and revise the content, and improve their understanding of the subject matter. The proposed system offers a user-friendly interface that allows students to interact with the system easily, ensuring a seamless experience. The existing methodology for speech-to-text conversion provides a foundation for the project. Techniques such as automatic speech recognition and language modeling have been extensively researched and implemented to achieve high accuracy in transcribing speech. The project builds upon these existing methodologies and adapts them to cater specifically to the needs of vernacular students.

The proposed methodology leverages the power of NLP techniques to enhance the quality and coherence of the transcriptions. Tokenization breaks down the text into manageable units, while part-of-speech tagging provides insights into the syntactic structure. Named entity recognition identifies important entities, and dependency parsing captures grammatical relationships between words. Sentiment analysis helps understand the emotional tone, and language modeling improves the overall accuracy and contextuality of the transcriptions. Data preprocessing plays a critical role in ensuring the quality of the input. Techniques such as noise removal, normalization, segmentation, and feature extraction are employed to enhance the speech signal and prepare it for accurate recognition. The project also incorporates performance optimization techniques to ensure real-time processing of the audio input, enabling efficient and seamless conversion of speech to text.

In conclusion, the speech-to-text conversion project offers a solution that enables vernacular students to convert speech into text, facilitating their understanding and engagement with study materials. The proposed methodology, leveraging NLP techniques and optimizing performance, ensures accurate and coherent transcriptions. By implementing this project, we can bridge the gap and provide equal opportunities for vernacular students, enabling them to overcome their challenges and thrive in their educational pursuits.

6.2 FUTURE ENHANCEMENT

The speech-to-text conversion project has significant potential for future enhancements and improvements. Some possible avenues for future development include:

Multilingual Support: Expanding the system to support multiple languages would make it more versatile and accessible to a broader range of users. By incorporating language-specific models and resources, the system can accurately transcribe speech in different languages, catering to a more diverse user base.

Real-time Streaming: Currently, the system processes pre-recorded audio input. Introducing real-time streaming capabilities would enable users to convert speech to text in real-time, opening up possibilities for live transcription services, online communication platforms, and real-time captioning for live events.

Adaptive Language Models: Developing adaptive language models that can learn from user interactions and adapt to specific domains or user preferences would enhance the accuracy and contextuality of the transcriptions. This would allow the system to better understand and transcribe specialized vocabulary or subject-specific jargon.

Speaker Diarization: Incorporating speaker diarization capabilities would enable the system to identify and distinguish between multiple speakers in a conversation. This would be particularly useful in scenarios such as transcribing meetings, interviews, or group discussions where multiple speakers are involved.

Semantic Analysis: Integrating semantic analysis techniques would enhance the understanding of the transcribed text beyond syntactic and grammatical information. This would enable the system to extract deeper meaning, identify relationships between concepts, and provide more insightful analysis of the content.

Interactive and Contextual Dialogue: Developing interactive dialogue systems that can engage in contextually relevant conversations with users would enhance the user experience. This could involve incorporating chatbot functionalities, allowing users to ask questions, seek clarifications, or request additional information based on the transcribed text.

Improved Accuracy and Robustness: Continuously improving the accuracy and robustness of the speech recognition models through advanced machine learning and deep learning techniques is a crucial aspect. Incorporating state-of-the-art models, exploring transfer learning approaches, and leveraging larger and more diverse training datasets would help achieve higher transcription accuracy.

Integration with Assistive Technologies: Integrating the speech-to-text conversion system with assistive technologies, such as screen readers or text-to-speech systems, would provide a comprehensive solution for individuals with visual impairments or reading difficulties. This would enable them to access and interact with the transcribed text effectively.

Crowdsourcing and Feedback Mechanisms: Implementing crowdsourcing and feedback mechanisms would allow users to contribute to the improvement of the system. Users could provide feedback on transcriptions, correct errors, and provide annotations to enhance the quality of the training data, leading to continuous refinement and improvement of the system.

Integration with Educational Platforms: Integrating the speech-to-text conversion system with educational platforms and e-learning tools would enable seamless integration into existing educational workflows. This would allow teachers to generate transcriptions for educational videos, enable interactive exercises based on transcribed content, and provide personalized learning experiences.

CHAPTER 7
REFERENCE

1. Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., ... & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine, 29(6), 82-97.
2. Graves, A., Mohamed, A. R., & Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 6645-6649).
3. Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (3rd ed.). Pearson. Li, X., Chen, X.,
4. Li, L., Xu, X., & Wu, X. (2018). A survey of deep learning techniques for automatic speech recognition. ACM Transactions on Intelligent Systems and Technology (TIST), 9(2), 1-27.
5. Young, S., Evermann, G., Gales, M. J., Hain, T., Kershaw, D., Liu, X., ... & Woodland, P. C. (2006). The HTK book (version 3.4. 1).
6. Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Transactions on Audio, Speech, and Language Processing, 20(1), 30-42.
7. Zoraida, F. R., Neumeyer, L., Cieri, C., & Phillips, J. D. (2002). The conversational telephone speech (CTS) collection. In Proceedings of the International Conference on Language Resources and Evaluation (LREC).
8. Chiu, C. C., & Wu, C. H. (2016). State-of-the-art speech recognition with sequence-to-sequence models. In Proceedings of INTERSPEECH (pp. 480-484).
9. Renals, S., Gales, M. J., Woodland, P. C., Pye, D., & Young, S. J. (2000). The HTK large vocabulary speech recognition system: User guide (version 3.2).
10. Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016). Character-aware neural language models. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence.
11. Collobert, R., & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine Learning (pp. 160-167).
12. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
13. Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems .

APPENDIX -1

SOURCE CODE

```
 import numpy as np
import
glob
import
os
import argparse
imagestore=[]
! pip install -q transformers
import librosa
import torch
from transformers import
Wav2Vec2ForCTC,Wav2Vec2Tokenizer
#load pre-trained model and tokenizer
tokenizer =
Wav2Vec2Tokenizer.from_pretrained("facebook/wav2vec2-
base-960h")
model =
Wav2Vec2ForCTC.from_pretrained("facebook/wav2vec2-
base-960h")

#load any audio file of your choice
speech, rate = librosa.load("batman1.wav",sr=16000)

import IPython.display as display
display.Audio("batman1.wav", autoplay=True)

input_values = tokenizer(speech, return_tensors =
'pt').input_values

input_values

#Store logits (non-normalized predictions)
```

```
logits = model(input_values).logits

#Store predicted id's
predicted_ids = torch.argmax(logits, dim =-1)

#decode the audio to generate text
transcriptions = tokenizer.decode(predicted_ids[0])
print(transcriptions)
from SpatioTempralModel import loadModel import numpy as
np
import argparse
parser=argparse.ArgumentParser()
parser.add_argument('n_epochs',type=i
nt) args=parser.parse_args()
X_train=np.load('training.npy')
frames=X_train.shape[2]
#Need to make number of frames divisible by 10
frames=frames-frames%10 X_train=X_train[:,:,:frames]
X_train=X_train.reshape(-1,227,227,10)
X_train=np.expand_dims(X_train,axis=4)
Y_train=X_train.copy()
epochs=args.n_epo
chs batch_size=32
if name =="_main__":
        model=loadModel()
        callback_save = ModelCheckpoint("model.h5",
                                monitor="mean_squared_error",
        save_best_only=True) callback_early_stopping =
        EarlyStopping(monitor='val_loss', patience=3) print('Model has been
        loaded')
        model.fit(X_train,Y_train,

        batch_size=batch_size,
        epochs=epochs,
        callbacks = [callback_save,callback_early_stopping]
        )
```

Splitting the video:

```
import os

import subprocess

def vid_into_frames(root_dataset, out_path):

    for i, (dire, folds, fil) in
        enumerate(os.walk(root_dataset)): for vid_name in
        fil:
            # path to video

            vid_path = os.path.join(dire,
            vid_name) # create a folder for each
            video
            fname = vid_name.split('.avi')[0]

            path_fold = os.path.join(out_path, dire.split('/')[-1],
            fname) if not os.path.isdir(path_fold):
                os.makedirs(path_fold)

            # executing ffmpeg -i file.mp4 -vf fps=5 path/%04d.jpg
            print("#################################################
            #")
            print(path_fold)

            cmd = "ffmpeg -i {} -vf fps=5 {}/%04d.jpg".format(vid_path, path_fold)
            subprocess.call(cmd, shell=True)
```

APPENDIX -2

IMPLEMENTATION AND RESULT

INPUT:

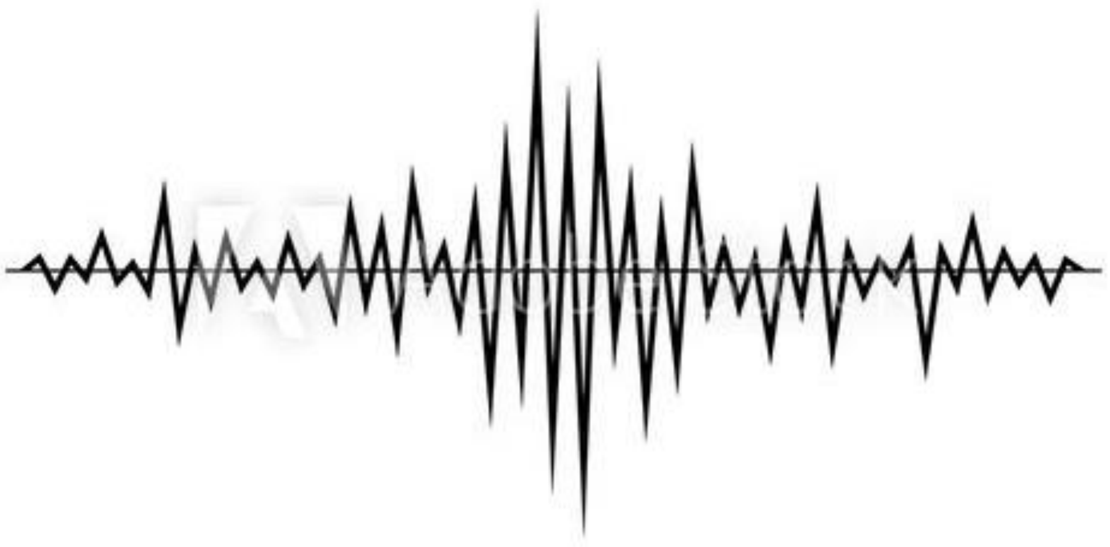
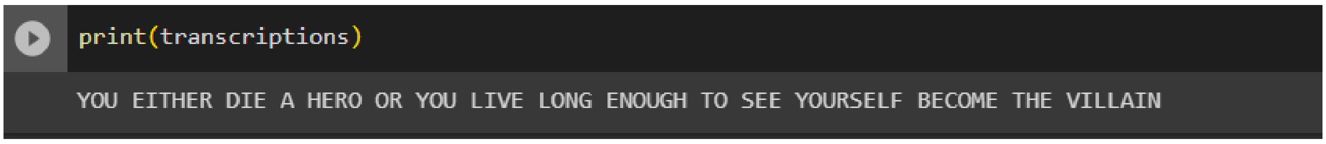


Figure A 2.1 Input Audio

Audio input refers to the process of capturing sound or spoken language as an input for a system or application. In the context of speech-to-text conversion, audio input refers to the spoken words or sounds that need to be converted into written text using natural language processing (NLP) techniques.

Audio File Input: Audio input can also be in the form of pre-recorded audio files, such as WAV, MP3, or other common audio formats. These files can be stored locally on a device or retrieved from a remote server or database. The audio file is then processed to extract the speech and convert it into text.

It's important to note that audio input may present certain challenges, such as background noise, varying audio quality, different speaking styles or accents, and potential speech recognition errors. Pre-processing techniques, such as noise reduction or audio normalization, may be employed to improve the accuracy of the speech-to-text conversion. The audio input is a crucial component in the speech-to-text conversion process as it serves as the raw material for extracting spoken words. **OUTPUT:**



```
print(transcriptions)
```

```
YOU EITHER DIE A HERO OR YOU LIVE LONG ENOUGH TO SEE YOURSELF BECOME THE VILLAIN
```

Figure A 2.2 Output as Text

Speech-to-text using natural language processing (NLP) is a technology that converts spoken language or audio input into written text. It combines the power of automatic speech recognition (ASR) and NLP techniques to transcribe spoken words into a textual format. The process of speech-to-text using NLP involves several steps. First, the audio input, such as recorded speech or real-time audio, is captured. Then, the speech signal is processed using ASR algorithms to convert the analog sound waves into digital representations. This involves techniques like signal preprocessing, feature extraction, and acoustic modeling. Once the speech is converted into a digital format, NLP techniques are applied to interpret and convert it into written text.

NLP algorithms analyze the linguistic aspects of the speech, including word segmentation, part-of-speech tagging, syntactic parsing, and semantic analysis. These techniques help in understanding the structure and meaning of the spoken language. NLP also incorporates language models and statistical methods to improve the accuracy of the transcriptions. This involves predicting the most probable sequence of words based on the context and language patterns. NLP algorithms may utilize machine learning approaches, such as recurrent neural networks (RNNs) or transformer models, to capture the dependencies and nuances of the language. The output of speech-to-text using NLP is the converted text representation of the spoken words.

This output can be further utilized in various applications, including transcription services, voice assistants, language translation, sentiment analysis, and information retrieval. Overall, speech-to-text using NLP enables efficient and accurate conversion of spoken language into written text, facilitating accessibility, information retrieval, and language processing tasks. It has significant implications in various domains, ranging from education and healthcare to communication and automation.