Reportlab Based Transcription Transformer

P.Anjaneyulu, D. Priyanka, Y. Manaswini, G. Sanjana, B. Sai Haneesha, B. Manideep, M. Harshitha

Department of Computer Science & Engineering Aditya Institute of Technology and Management, Tekkali, India, 532201

Corresponding author: P.Anjaneyulu, Email: anjii.ram888@gmail.com

In an era where digital communication is paramount, harnessing the power of voice data and converting it into text and structured documents is unprocurable. This system excels at converting Audio files into text and seamlessly transforming text into PDF documents, with the overarching goal of providing a seamless and comprehensive solution for converting spoken words into written text and subsequently crafting polished PDF documents. The core of this lies in implementing advanced speech recognition algorithms to transcribe spoken words accurately. Leveraging machine learning and natural language processing techniques, it achieves high precision in converting audio input into text format. Furthermore, it offers flexibility by accommodating language and accents. Once the speech is accurately transcribed, the system seamlessly integrates with PDF generation capabilities. It assembles the transcribed text into well-structured PDF documents, complete with formatting options. The significance can hold immense potential to revolutionize transcription services, offering a groundbreaking solution that drastically reduces the time and effort required by professionals in various domains. It improves accessibility for differently-abled individuals in healthcare only aids individuals with hearing impairments. and education in Lecture Transcription and Note- Taking. Its user-friendly interface makes it valuable in many contexts.

Keywords: Long Short-Term Memory, Convolutional Neural Network, Transformers, ReportLab, FPDF(F-PDF).

2025. In Sandeep Kumar & Kavita Sharma (eds.), *Computational Intelligence and Machine Learning*, 85–96. Computing & Intelligent Systems, SCRS, India. DOI: https://doi.org/10.56155/978-81-975670-5-6-7

1 Introduction

The research focuses on harnessing the potential of voice data by seamlessly converting audio files into polished PDF documents. The key strength lies in the implementation of advanced speech recognition algorithms utilizing machine learning techniques for highly accurate transcription. This system not only ensures precision but also accommodates various accents for enhanced flexibility. If there is any pause in the audio, the system intelligently starts a new paragraph, enhancing the readability of the transcribed text. Moreover, if any sentence is repeated more than once in the audio, it is automatically highlighted, bringing attention to potential emphasis. Once the spoken words are accurately transcribed, the system seamlessly integrates with PDF generation capabilities, creating well-structured documents with formatting options. Tensor Flow simplifies creating, training, and deploying speech recognition models, making complex tasks easier for developers with user-friendly tools and optimization algorithms.

2 Literature Survey

Rishabh Jain et al. explored [1], the objective is to improve automatic child speech recognition by experimenting with different pretraining and fine tuning configurations of the ASR model, wav2vec2, using self-supervised learning (SSL). The goal is to determine the ideal amount of data required to effectively fine- tune the model for child ASR tasks.

Zhenzi Weng et al. developed [2], The Deep SC-ST introduces a semantic communication system driven by deep learning for speech transmission. It demonstrates outstanding performance in low signal-to-noise ratio environments when contrasted with traditional communication systems and contemporary deep learning-enabled alternatives.

Jane Oruh et al. proposed [3], a refined LSTM recurrent neural network (RNN) model using deep learning for automatic speech recognition (ASR) achieved enhanced performance compared to CNN-based and sequential models.

Ali Bou Nassif et al. provided [4], Since its inception in 2006, deep learning has undergone extensive exploration within speech applications. This burgeoning field of machine learning has consistently outperformed other methodologies across diverse applications, particularly in the realm of speech. The insights presented in this paper provide a thorough examination of research conducted in this domain, elucidating trends and pointing towards emerging areas of interest.

Akafumi Moriya et al. proposed [5], a neural transducer architecture underpins an end-to-end target-speaker speech recognition system, enabling streaming and on-device recognition. This system is designed to discern when the target speaker is inactive and output no response accordingly. It competes effectively with traditional cascade approaches, combining target speech extraction and recognition modules, while also lowering computational expenses and facilitating streaming decoding.

Yash Agrawal et al. illustrates [6], captions are generated from online meetings and linked to the corresponding meeting transcript, which is then sent to the backend for summarization using an NLP model.

Tanvirul Alam et al. explored [7], Various transformer-based models are explored along with proposing an augmentation strategy for the punctuation restoration problem, with a focus on both high-resource (English) and low-resource (Bangla) languages. The Bangla dataset developed by the team has been made publicly accessible to the research community.

Table 1. Study of past research papers

Author(s)	Year	Algorithm Used	Output
Moriya,T. et al.[9]	2023	Automatic Speech Recognition.	Transcribing text and detecting activities.
Kanda, F. G. et al.[10]	2023	Electroencephalogram	Freeware software for neurophysiologic Reports
Liya, B. S. et al.[11]	2023	Deep learning	Real-time audio summarization
Agrawal, Y. et al.[12]	2021	NLP models for Google Meet Transcript Summarization.	Summarized transcript
Basystiuk,O. et al.[13]	2021	Recurrent neural networks	Automatic audio to text conversion
Alam, T. et al.[14]	2020	Transformer models	Punctuation restoration in high and low-resource languages
Aouani, H. etal.[15]	2020	Deep learning	Speech emotion Recognition
Sherstinsky, A.[16]	2020	RNN and LSTM networks	Fundamentals of RNN and LSTM networks

3 Methodology

- **Convolutional Neural Network:** Its multifaceted advantages make it a crucial component in this speech recognition system [1]. Initially, CNNs are adept at extracting intricate features from audio spectrograms, enabling the model to discern crucial frequency patterns inherent in speech. Secondly, by capturing temporal dependencies in the data, facilitate the understanding of sequential information, essential for recognizing spoken language nuances.

Additionally, contribute to the model's efficiency by reducing the complexity of the input representation, streamlining the learning process and enhancing overall accuracy.

Eventually, the use of shared weights in convolutional layers allows the network to generalize and learn diverse features, making it well-suited for robust speech pattern recognition across varied inputs [1][2].

$$O_{i,j} = f\left(\sum_{i=1}^{M-1}\sum_{j=1}^{N-1} I_{i+m,j+n} \times K_{m,n} + b\right) \quad (1)$$

$O_{i,j}$ output at position (i,j)

$I_{i+m,j+n}$ is the input at position (i+m,j+n)

$K_{m,n}$ is the convolutional filter

b is the bias term, f is the activation function

The formula represents a convolutional neural network (CNN) operation where each output pixel Oi,j is computed by convolving input pixels I with a filter K, adding a bias term, and applying an activation function f.

- **ReportLab:** ReportLab is like a virtual printer for Python. ReportLab is helpful for converting text to PDF documents because it allows you to control the layout, fonts, colors, and styles of the text in your Python code [3]. It's a versatile tool for making customized PDF reports or documents with text content [4].
- **Speech Recognition:** The Speech Recognition module in Python offers developers a versatile platform for recognizing speech from various audio sources, leveraging multiple recognition engines such as Google SR, Sphinx, Wit.ai, and Microsoft Azure Speech. Its user-friendly API simplifies integration, facilitating rapid incorporation of speech recognition capabilities into Python applications. Capable of processing diverse audio sources including microphone input, audio files, and streams, Speech Recognition ensures accurate recognition results, especially with cloud-based engines like Google SR. It supports both offline and online recognition modes, providing flexibility for different application scenarios. Moreover, developers have the option to customize and configure recognition settings to suit specific requirements and use cases, enhancing the module's adaptability and functionality [5].
- **Long Short-Term Memory:** This is valuable for capturing context and sequential patterns in audio data [6]. LSTMs retain memory of previous inputs, enabling the model to understand temporal dependencies crucial for speech recognition [7]. This helps in recognizing and interpreting spoken words within the context of the entire audio sequence, enhancing the accuracy of the speech recognition model [8].

$$f_t = \sigma g\big(W_f[ht-1, xt] + bf\big) \quad f_t \ \textit{is the forget gate}$$

$$i_t = \sigma g(W_i[ht-1, xt] + bi) \quad i_t \ \textit{is the input gate}$$

$$\underset{C_t}{\rightarrow} = tanh(W_c[ht-1, xt] + bc) \quad \underset{C_t}{\rightarrow} \ \textit{is the candidate cell state}$$

$$C_t = ft * Ct - 1 + it * \underset{C_t}{\rightarrow} \quad C_t \ \textit{is the Cell State}$$

$$o_t = \sigma g(W_o[ht-1, xt] + b_o) \quad o_t \ \textit{is the output gate}$$

$$h_t = o_t * \tan h(C_t) \quad h_t \ \textit{is the hidden state}$$

- F-PDF: FPDF is useful for converting text to PDF, You can use it to add text, format it, and create pages in a PDF document.
- CTC Loss: CTC LOSS helps the speech recognition model learn by allowing it to handle different lengths of spoken words and match them to transcriptions, even when the timing is not perfect. It's like giving the model flexibility to understand words, even if they're said at different speeds or with pauses.

4 Existing System

The existing system uses Automatic Speech Recognition; ASR performance can degrade in noisy environments. Background noise, accents, or variations in speaking styles may hinder accurate transcription. ASR models may struggle when dealing with diverse accents, speech rates, or individual speaking habits that differ from the training data. It might have difficulty recognizing words or phrases do not present in their training data. This can be a challenge for handling new vocabulary.

In digital communication, Speech-to-Text and later structuring it into PDF documents with different formatting options is unavailable.

Limitations:

- Noise Sensitivity
- Less Readability
- No Structuring of PDF document

5 Proposed System

In this, we want to use Speech Recognition, Reportlab, CNN, LSTM. CNN is key for speech recognition, extracting vital features from audio for understanding frequency patterns and sequential information. It boosts accuracy by reducing input complexity. If there is any pause in the audio, the system intelligently starts a new paragraph, enhancing the readability of the transcribed text. Moreover, if any sentence is repeated more than once in the audio, it is automatically highlighted, bringing attention to potential emphasis. Once the spoken words are accurately transcribed, the system seamlessly integrates with PDF generation capabilities, creating well- structured documents with formatting options. ReportLab and F-PDF are used to convert the recognized text to PDF document. This system not only ensures precision but also accommodates various accents for enhanced flexibility.

- More Readability
- Seamless Conversion of Audio file into PDF Document

6 Problem Description

The problem at hand is to design and implement a solution for harnessing the power of voice data and converting it into text and structured documents. Basically, in an era where digital communication is paramount but currently unprocurable. This proposed system excels at converting audio files into text and seamlessly transforming text into PDF documents, with the overarching goal of providing a seamless and comprehensive solution for converting spoken words into written text and subsequently crafting polished PDF documents. The core of this lies in implementing advanced speech recognition algorithms to transcribe spoken words accurately. Leveraging Machine Learning and Deep Learning techniques, it achieves high precision in converting audio input into text format. Furthermore, it offers flexibility by accommodating language and accents. Once the speech is accurately transcribed, the system seamlessly integrates with PDF generation capabilities. It assembles the transcribed text into well-structured PDF documents, complete with formatting options.

7 Implementation

Audio File: Audio-to-PDF generation involves converting audio to text through speech-to-text technology. The transcribed text is processed for accuracy and formatting, then structured into a PDF using programming languages like Python with tools such as ReportLab or PDFKit. The PDF may include formatting, metadata (title, author, date), and additional features like bookmarks. Quality checks ensure accuracy, and the finalized PDF serves as a container for the transcribed content. It is suitable for storage, distribution, and review, providing a standardized and versatile format for audio-to-text applications.

Code:

1 Enabling cross-platform utilization of operating system-specific functionalities are used to import the 'os' module. Import the `speech recognition` module and aliases it as `sr` for easier reference. Import the `canvas` class from the `reportlab.pdfgen` module, which allows creating PDF documents. Import the `A4` constant from the`reportlab.lib.page sizes` module, which represents the dimensions of an A4 page. Imported all the required Libraries as shown in Figure 1.

```
import os
import speech_recognition as sr
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4
```

Figure 1. Libraries Used

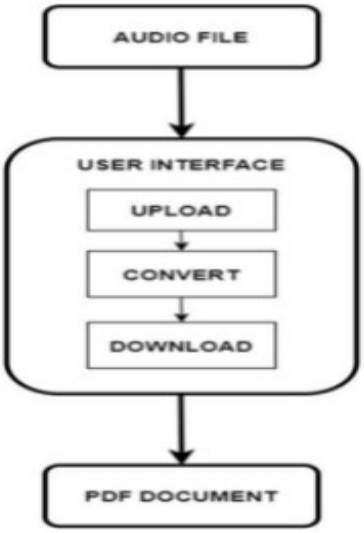

Figure 2. System Architecture

2 Defines a function named `convert_wav_to_text` that takes a WAV file path (`wav_file`) as input. Creates an instance of the `Recognizer class from the`speech_recognition` module for recognizing speech. Opens the WAV file specified by `wav_file` using the `Audio File` class from the`speech_recognition` module and assigns it to `source`. Records the audio data from the `source` using the `record` method of the `Recognizer` instance. Begins a try-except block to handle potential errors during speech recognition. Uses Google's speech recognition service to recognize the speech from the audio data and stores the result in the `text` variable as discussed in Figure 2. Returns the recognized text. Except sr. Unknown Value Error: Handles the case where the speech recognizer is unable to understand the audio. Except sr .Request Error as e: Handles the case where there is an error while making a request to Google's speech recognition service shown in Figure 3.

```
def convert_wav_to_text(wav_file):
    try:
        recognizer =  sr.recognizer()
        with sr.audiofile(wav_file) as source:
            audio_data = recognizer.record(source)
            text = recognizer.recognize_google(audio_date)
            return text
        except sr.unknownvalueerror:
            return "speech recognition could not understand audio"
        except sr.requesterror as e:
            return f
```

Figure 3. Code to Convert Wav to text

3 def create_pdf(text, pdf_file): Defines a function named `create_pdf` that takes the recognized text and the PDF file path as input.Pdf = canvas. Canvas(pdf_file, page size=A4): Creates a PDF canvas object with the specified file path (`pdf_file`) and A4 page size.pdf.setFont("Times-Roman", 12): Sets the font of the PDF to "Times-Roman" with a font size of 12 as discussed in Figure 4.

```
def create_pdf(text, pdf_file):

    pdf = canvas. canvas(pdf_file, pagesize=a4)

    pdf.setfont("times-roman", 12)
```

Figure 4. Using of pdf module

Splits the recognized text into lines using the newline character `\n` and assigns them to the `lines` variable. Initializes variables for positioning text in the PDF. Iterates over each line in the `lines` list. Splits each line into words. Constructs lines of text that fit within the width of the PDF page. Draws the constructed lines of text onto the PDF canvas at specified positions. pdf.save(): Saves the PDF document. Figure 5 shows the code implementation for saving audio files as PDFs and the process of saving the generated PDFs.

```
for line in lines:
    words = line.split()
    current_line = words[0]
    for word in words[1:]:
        if  pdf.stringwidth(current_line+" " +word, "times-roman", 12) < max_width:
            current_line +=" "+word
        else:
            pdf.drawstring(72, y_position,current_line)
            y_position -= line_height
            current_line = word
     pdf.drawstring(72, y_position, current_line)
     y_position -= line_height
pdf.save()
```

Figure 5. Illustrates the code implementation saving audio files as PDFs and the process of saving the generated PDFs.

Upload: Interface has an "Upload" button for users to select audio (HTML + server-side tech). Server receives, validates, and possibly converts the audio file. Server uses libraries/tools to generate a PDF with embedded audio.

Convert: Interface offers a "Convert" option for users to select and convert audio files. Server receives the audio file, processes conversion (if needed), ensuring compatibility. Server generates a PDF with the converted audio, utilizing appropriate tools or libraries.

Download: Interface includes a "Download" option for users to obtain the generated PDF. If not done earlier, server may finalize PDF generation based on user preferences. User is presented with a link/button to download the completed PDF file. Once downloaded, the user can view or share the PDF containing the audio.

8 Flowchart

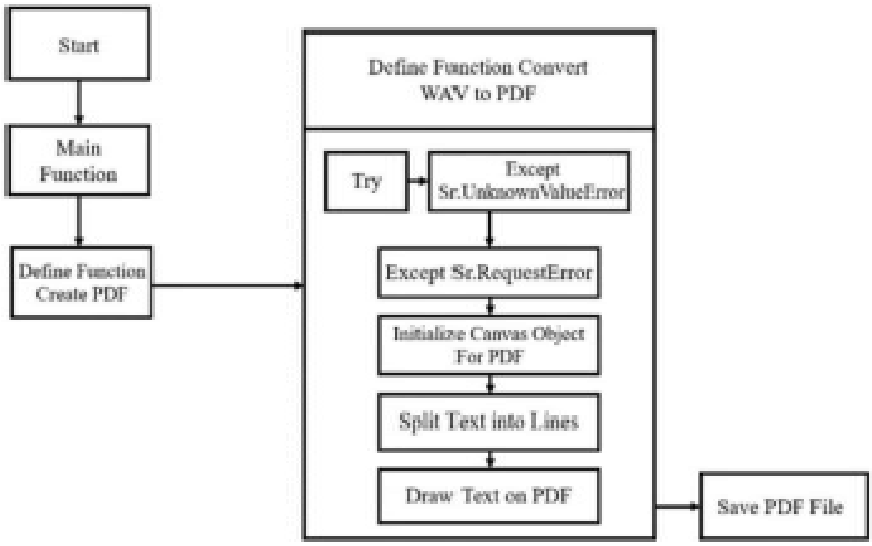

Figure 6. Flowchart

The above Figure 6 shows the flowchart attempts to convert a WAV file to PDF using the "sr" library. If unsuccessful, it falls back on creating a PDF with extracted text using the "pypdf2" library.

Defined functions "convert_wav_to_pdf" handles the conversion with error handling, and "create_pdf" processes and formats the text content for the PDF. The "Main Function" initiates the conversion and saves the generated PDF. While this offers a simplified overview, it might not encompass all program intricacies.

9 Result and Analysis

Users upload a WAV file by clicking the "Browse" button and selecting the file from their computer as shown in Figure 7. Once the file is selected, users can click the "Generate Pdf" button to start the conversion process. The website uses a library called Speech Recognition to convert the WAV file to

text. Once the conversion is complete, the website creates a PDF file and saves it to the user's computer as shown in Figure 8. The user can then click the "Download" button to download the file.

Figure 7. Login page

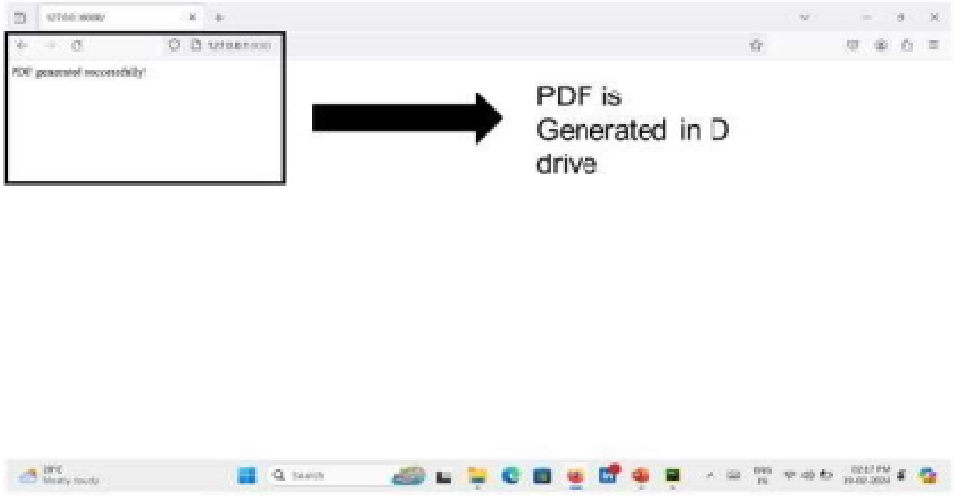

Figure 8. PDF Downloaded in D drive

Average Accuracy:

Table 2. Speech Recognition Accuracy by Audio Length

Seconds	No. of Audio Clips	Average Accuracy
10	25	95.8
20	22	94.2
30	20	93
40	19	91.7
50	17	88.76
60	16	87.7
70	15	85
80	13	82.45
90	11	80.5
100	8	78.76
110	7	75.12
120	6	75
130	6	72.15
140	5	71.5
150	4	70

The provided table 2 shows higher accuracy for shorter clips and lower accuracy for longer ones. This suggests that processing a larger amount of audio data poses a challenge for the speech recognition system. See Figure 9.

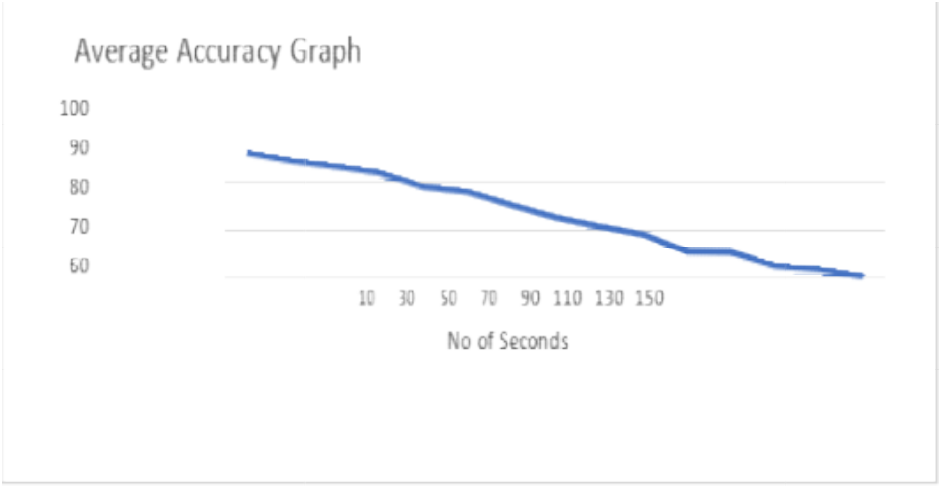

Figure 9. Speech Recognition Accuracy vs. Audio Duration

10 Conclusion

This paper represents a groundbreaking leap in the era of digital communication, addressing the paramount need to convert voice data into structured text and documents. With a focus on seamlessly converting audio files to text and crafting polished PDF documents. The potential impact of this is vast, offering a transformative solution to revolutionize transcription services across diverse professional domains. By significantly reducing time and effort, it promises efficiency for professionals. Beyond its practical applications, contributes to improving accessibility for differently-able individuals, particularly those with hearing impairments, in healthcare and education contexts like Lecture Transcription and Note- Taking. The user-friendly interface further enhances its value, making it a versatile asset across multiple contexts in the digital landscape. In essence, this research work stands as a beacon of innovation, poised to shape the future of voice data conversion and document creation.

11 Future Scope

Enhanced Readability Features: Implement algorithms to detect natural pauses in the audio, automatically introducing new paragraphs for improved document structure, ensuring a more readable output. Intelligent Sentence Detection: Develop functionality to identify repeated sentences within the audio transcription, incorporating a highlighting mechanism in the generated PDF to draw attention to duplicated content. Advanced Language Support: Expand language and accent recognition capabilities to include a broader range of linguistic nuances, making the system adaptable to diverse communication styles and linguistic variations.

Reference

[1] Jain, R., Barcovschi, A., Yiwere, M., Bigioi, D., Corcoran, P., & Cucu, H. (2023). A Wav2Vec2-Based Experimental Study On Self-Supervised Learning Methods To Improve Child Speech Recognition. IEEE Access.

[2] Weng, Z., Qin, Z., Tao, X., Pan, C., Liu, G., & Li, G. Y. (2023). Deep learning enabled semantic communications with speech recognition and synthesis. IEEE Transactions on Wireless Communications.

[3] Damiani, C., Rovida, L., Maspero, D., Sala, I., Rosato, L., Di Filippo, M., ... & Mauri, G.(2020). MaREA4Galaxy: Metabolic reaction enrichment analysis and visualization of RNA-seq data within Galaxy. Computational and structural biotechnology journal, 18, 993-999.

[4] Pritchard, L., White, J. A., Birch, P. R., & Toth, I. K. (2006). GenomeDiagram: a python package for the visualization of large-scale genomic data. Bioinformatics, 22(5), 616-617.

[5] Nassif, A. B., Shahin, I., Attili, I., Azzeh, M., & Shaalan, K. (2019). Speech recognition using deep neural networks: A systematic review. IEEE access, 7, 19143- 19165.

[6] Oruh, J., Viriri, S., & Adegun, A. (2022). Long short-term memory recurrent neural network for automatic speech recognition. IEEE Access, 10, 30069- 30079.

[7] Behera, R. K., Jena, M., Rath, S. K., & Misra, S. (2021). Co-LSTM: Convolutional LSTM model for sentiment analysis in social big data. Information Processing & Management, 58(1), 102435.

[8] Smagulova, K., & James, A. P. (2019). A survey on LSTM memristive neural network architectures and applications. The European Physical Journal Special Topics, 228(10), 2313- 2324

[9] Moriya, T., Sato, H., Ochiai, T., Delcroix, M., & Shinozaki, T. (2023). Streaming End-to-End Target-Speaker Automatic Speech Recognition and Activity Detection. IEEE Access, 11, 13906-13917.

[10] Kanda, F. G., Salem, L. H., Kanda, R. G., & Kanda, P. A. M. (2023). EEG Weaver Reporter. A Freeware software for neurophysiologic reports. Revista Neurociências, 31, 1-13.

[11] Liya, B. S., Seenuvasan, V., Sathya Prakash, K., & Siva, B. (2023). Audio summarization in real time for podcast, speeches and audio books. Journal of Survey in Fisheries Sciences, 10(4S), 2618-2625.

[12] Agrawal, Y., Thakre, A., Tapas, T., Kedia, A., Telkhade, Y., & Rathod, V. (2021). Comparative analysis of NLP models for Google Meet Transcript summarization. EasyChair Preprint,(5404).
[13] Basystiuk, O., Shakhovska, N., Bilynska, V., Syvokon, O., Shamuratov, O., & Kuchkovskiy, V. (2021). The Developing of the System for Automatic Audio to Text Conversion. In IT&AS (pp. 1-8).
[14] Alam, T., Khan, A., & Alam, F. (2020, November). Punctuation restoration using transformer models for high-and low-resource languages. In Proceedings of the Sixth Workshop on Noisy User-generated Text (W NUT 2020) (pp. 132-142).
[15] Aouani, H., & Ayed, Y. B. (2020). Speech emotion recognition with deep learning. Procedia Computer Science, 176, 251-260.
[16] Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306.