International Conference on Recent Trends in Data Science and its Applications
DOI: rp-9788770040723.141

Audio Summarization in Real Time for Podcasts

S. Jeeva
Assistant Professor.
Department of Data Science and Business Systems,
SRM Institute of Science andTechnology,
Kattankulathur, Chennai, India
jeevas@srmist.edu.in

GudlaSaiSujan
UG Student, B.Tech.
Department of Data Science and Business Systems,
SRM Institute of Science andTechnology,
Kattankulathur, Chennai, India
Gs8426@srmist.edu.in

ArutlaSiddharth Reddy
UG Student, B.Tech.
Department of Data Science and Business Systems,
SRM Institute of Science andTechnology,
Kattankulathur, Chennai, India
ar7613@srmist.edu.in

***Abstract*—The COVID-19 pandemic has led to a significant shift towards online work for various activities such as education, job interviews, healthcare consultations, and company meetings. This has resulted in the widespread use of online meeting software applications like Google Meet and Microsoft Teams. The practical applications of this topic are of great importance and relevance in the current scenario.This paper presents a comprehensive approach to generating text summaries from both text and voice audio files. In the first scenario, the audio files are converted to text format using Python tools, while in the latter scenario, text summarization is performed using modules from Natural Language Processing. Specifically, the SpaCy Python framework is utilized for English data functions. The significant sentences identified during the text extraction process are used in the summarization approach. The weight assigned to each word is based on the number of times it appears in the text file. By starting with the main audio recording, this method is utilized to generate summaries, Using the main audio recording as a starting point, this method is used to create summaries.**

Overall, this approach provides a useful solution for extracting important information from both text and voice audio files. By leveraging advanced techniques such as Natural Language Processing and text extraction, the summarization process is able to identify key points and reduce the overall amount of information without sacrificing important details. The result is a concise and accurate summary that can be quickly and easily read, making it an ideal solution for busy professionals who need to process large amounts of information efficiently.

Keywords: Deep learning, python speech recognition, of TF-IDF, SpaCy, Natural language processing.

I. Introduction

Over the last two decades, the popularity of multimedia applications like gaming, virtual and augmented reality (VR and AR), teleconferencing, and entertainment has led to a surge in demand for immersive communication technologies. The proliferation of mobile multimedia and ubiquitous computing has played a crucial part in the development of these technologies. Spatial audio research is currently a key focus in this field, as it offers essential resources for capturing, processing, and reproducing spatial sound, which is essential for creating three-dimensional (3D) immersive user experiences.

Spatial audio, a research field that requires collaboration among specialists in areas such as audio engineering, acoustics, computer science, and applied psychoacoustics, aims to replicate or create new acoustic environments by utilizing suitable sound recording, processing, and reproduction techniques. Maintaining the precision of not only the audio content but also the spatial features of the sound scene, which depend on the physical positions of sound sources and acoustic characteristics of the environment, is crucial in achieving this objective.

The methods and techniques used in spatial audio can be divided into three stages: capture, processing, and reproduction. The first stage involves recording the sound scene, while the second stage involves modifying the recorded spatial information or extracting additional information that was not captured. The final stage involves reproducing the processed sound scene, resulting in a realistic auditory experience.

Summarization is the process of making data easier and quicker to comprehend while preserving the language's grammar and meaning. Optical character recognition (OCR), on the other hand, involves identifying text from digitized documents or images. When a PDF file is uploaded to our tool, it is first converted into an image array using the pdf2image Python module, which is a wrapper around the pdftoppm and pdftocairo command line tools. The OCR receives a list of images, and each image is converted to digitized text using a combination of model recognition and feature detection techniques. The OCR then returns the extracted text in a string format.

With the rise of Internet platforms like YouTube, there is an increasing need for multimedia summarization. The goal of automatic summarization is to produce a concise and informative version of the original content. In this article, we focus specifically on audio summarization, which involves creating a summary of an audio signal. There are three methods for generating an audio summary: using only audio functions, extracting text from the audio signal, and using textual methods to guide the summarization process. A hybrid approach that combines the first two methods is also possible.

There are advantages and disadvantages to each approach for audio summarization. Relying solely on audio features to create a summary is independent of transcription, but it can be problematic since the summary is based only on how things are said. Conversely, using textual methods to guide the summarization process can leverage the information in the text, resulting in more informative summaries. However, transcripts may not always be available. Combining audio functions and textual methods can enhance the quality of the summary, but both approaches also have their drawbacks.

The field of voice recognition is a multidisciplinary area of study that involves the use of various technologies to recognize and interpret spoken language, ultimately converting it into text format. This field involves the application of different fields such as computer science, signal processing, linguistics, and artificial intelligence. The process of voice recognition involves the use of speech recognition algorithms that analyze and interpret speech signals, breaking down the audio into small components such as phonemes, words, and sentences.When a user downloads an audio file, it is first pre-processed for transcription. There are three ways to create an audio summary: using only audio functions, extracting text from the audio signal, and a hybrid approach that combines both methods. Each approach has its own advantages and disadvantages. Using only audio features provides independence from transcription, but may not capture the intended meaning accurately. On the other hand, leading the summary with textual methods may produce more informative summaries, but transcripts may not always be available. Finally, combining audio functions and textual methods may improve the summary's quality, but it also has its drawbacks.

II. Literature Review

Previous research has focused on the classification, while data augmentation has been utilized in various other applications to enhance model accuracy. A summary of the existing literature in these domains is provided in this section.

[1] The problem of automatic summarization can be tackled in various ways. These solutions include unsupervised techniques, as well as graph-based approaches that employ ranking to organize input text in a graph. Additionally, neural methods exist which are discussed in more detail in the subsequent paragraph and employ graph traversal algorithms.

[2] Next, we explore studies related to summarizing extensive texts and scholarly articles, which are the dominant types of lengthy documents in the field of summarization. Lastly, we provide an overview of datasets used for summarization tasks, with a specific emphasis on datasets for summarizing academic articles

[3] Despite the significant increase in audiovisual data in the last decade, there is a lack of dedicated tools and software programs for audio summarization. While various studies have explored modules used in constructing audio summarizers, none have focused on developing and optimizing systems specifically for audio summaries.

[4] Graph-based models, which are commonly used in extractive summarization, utilize many inter-sentence and query-sentence interactions. LexRank assigns scores to each sentence in a graph of sentence similarity. Manifold ranking, applied by Wan and Xiao, The system utilizes connections between sentences, documents, and queries to facilitate processing. We model these relationships, along with token-level graph connections, and then aggregate them to create distributed sentence representations, with the exception of cross-document relationships.

[5] Graph-based models are widely used in extractive summarization and involve several interactions between sentences and queries. For instance, LexRank assigns scores to sentences and generates a similarity graph. Similarly, in manifold ranking,The system utilizes connections between sentences, documents, and queries to facilitate processingare considered. We also use a graph at the token level to model the above relationships, which is then combined to produce distributed sentence representations, except for cross-document relationships.

[6] As audio/visual data consumption increases, audio file management needs to become more advanced. A new technique, known as divide-and-conquer, is being explored to effectively summarize lengthy audio messages or snippets and extract important information. The method consists of three modules, namely Speech-to-Text Conversion, Text Summarization, and Text-to-Speech Conversion. The output of each module, except for Speech-to-Text Conversion, is used as input for the next module in the sequence. The audio file acts as input for the first module, while the last module converts the summarized text generated by the Text Summarization module into an audio file. A web application with a user interface created with Flask is used to facilitate the model's interaction with users.

[7] The current datasets for query-focused summarization are too small to be effective for training data-driven algorithms. However, manually constructing such a corpus is a time-consuming and resource-intensive process. To address this issue, researchers have proposed an approach to extract and summarize document sentences, which allows for a better match between the large model and the small benchmarks. Experiments conducted on three DUC benchmarks indicate that a pre-trained WikiRaf model has already achieved acceptable performance levels. Furthermore, with specific benchmark dataset optimization and data augmentation, the model outperforms strong comparison systems.

[8] With the growing amount of video content available, an automatic video summary can provide significant time-saving and learning benefits. It is becoming increasingly important to correctly navigate through the large amount of user-generated videos available. Video summary has the potential to extract informative frames from a film, and is therefore seen as a useful method for maximizing the information content. By leveraging text summarization and video mapping algorithms, it is possible to retrieve key video elements from subtitles and use them to generate a summary of the movie's content.

[9] The advent of the internet and social media platforms has led to an abundance of data in various formats, including text, audio, and video. However, it can be challenging for users to obtain an accurate overview or extract crucial information from these files. Users often seek a summary of the most pertinent information that can be quickly gleaned from the source files. To this end, automatic text summarization (ATS) is the only viable method for summarizing a single document or multiple documents to extract essential information. Unfortunately, current ATS systems often produce inadequate summaries and require significant time and resources to process large documents due to incorrect encoding.

[10] Video data is composed of two modalities, namely audio and vision. Multimodal learning, particularly audiovisual learning, has gained traction in recent years due

to its potential to enhance the performance of various computer vision tasks. However, existing video summarization methods mainly rely on visual information and do not utilize the audio modality. In this study, researchers suggest that incorporating audio information can aid in the comprehension of visual content and structure, ultimately improving the summarization process.

III. EXISTING SYSTEM

The current system has limited capabilities as it can only detect emotions in response to an audio input. It relies on a Python speech recognition library that may not be compatible with all inputs due to its dependencies. Furthermore, the system is incapable of analyzing a large dataset of audio files as it has not been trained on such data.

IV. PROPOSED SYSTEM AND ARCHITECTURE

The proposed real-time system analyzes audio inputs, extracts important keywords, and summarizes them into a shorter format. It is designed for audio books and podcasts, and can handle large inputs with good accuracy and efficiency. Our plan is to build a web application using Streamlit, a free and open-source framework for constructing Python apps with minimal lines of code. The app will prompt users to provide the podcast ID and generate a transcripted summary of the podcast. The proposed system is basically shown in the following flowchart in Figure 1.

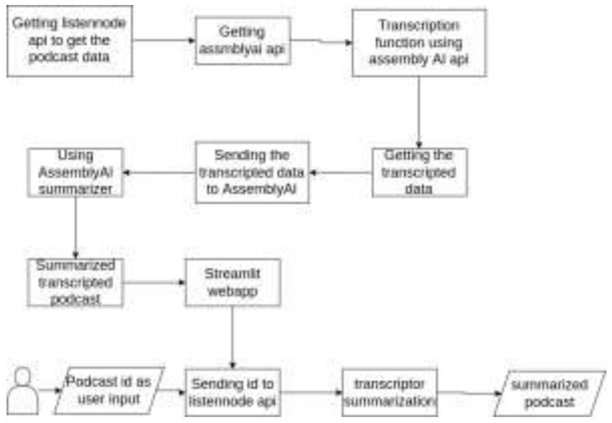
Getting listennode api to get the podcast data
Getting assmblyai api
Transcription function using assembly AI api
Getting the transcripted data
Sending the transcripted data to AssemblyAI
Using AssemblyAI summarizer
Summarized transcripted podcast
Streamlit webapp
Podcast id as user input
Sending id to listennode api
transcriptor summarization
summarized podcast

Fig. 1. Flowchart of the proposed system

V. IMPLEMENTATION

A. *Getting the PODCAST from the listennoteapi*

A method will be developed to retrieve podcast data based on the ID provided by the user as input.

In this section, we will be utilizing the Listen Notes API to extract the URL of the target podcast's episode. Listen Notes is a comprehensive online search engine and database for podcasts, providing access to podcast data through their API and enabling the creation of new applications and services based on it.

- Firstly, we need to create an account and subscribe to the free plan to access the data and utilize the Listen Notes API. The free plan allows a maximum of 300 requests per month, which is usually enough for personal projects.
- To proceed, we should visit the Listen Notes podcast page, choose the specific episode we want, and then click on the "Use API to get this episode" option.
- Afterwards, we can switch the language code to Python and choose 'requests' from the available options list, which will allow us to use the library in the future.
- After copying the code, ensure to paste it into your notebook or script.

Firstly, we use a GET request to access the Listen Notes Podcast API endpoint to retrieve the required information. The result is saved as a JSON object, containing the episode URL, which will be used later. Additionally, we import a JSON file named secrets.json, which is similar to a dictionary, containing key-value pairs. This file holds the API keys for both AssemblyAI and Listen Notes, and logging into your accounts is required to access them.

B. *Transcription and summarization of the podcast*

After reading the dataset, images will be preprocessed.in following steps,In this upcoming section, we will make use of a POST request instead of the previous GET request. Specifically, we will send a request to the transcript endpoint of the AssemblyAI API to request transcription. Uploading the audio URL to Assembly AI requires the use of the post method. Setting the auto chapter value to True is necessary in order to receive both the transcription and summary. If we set the auto chapter value to False, we would only be able to receive the transcription. Once this step is completed, we will store the ID of the transcription response.

Retrieve Transcription and Summary:

In the final step, we can retrieve the transcription and summary by sending a GET request to AssemblyAI. It may take several requests until the status of the response is completed. After that, we store the results in two separate files: a txt file for the transcription and a JSON file for the summary.

Transcription:

Converting an audio file into a text file is known as audio transcription, which can be applied to various situations such as interviews, academic studies, music video clips, and conference recordings. The AssemblyAI API will be utilized to transcribe the data fetched from the Listen Notes API and provide the transcription data.

Real-Time Streaming Transcription:

The Real-Time Streaming WebSocket API provides clients with text transcriptions in just a few hundred milliseconds through a streaming process. In cases where there is an error, the API will consistently return a JSON response.

Summarization:

Audio summaries are concise versions of audiobooks that effectively capture the key ideas and themes of longer works in a more accessible format. They provide a comprehensive overview of the author's content, style, and spirit. To retrieve the data that has been transcribed and summarized from the podcast, we will use the AssemblyAI API, which will be employed to transmit the data.

C. Web App

Our goal is to develop a web application using Streamlit that requests the user to enter a podcast ID and then provides a summarized transcription of the podcast. Streamlit is a Python framework that enables the creation of applications with minimal code. We begin by importing Python libraries and defining functions that replicate the aforementioned actions. Additionally, we provide a zipped package containing both the transcription and summary files. To start, we utilize st.markdown to display the application's main title. We create a left panel sidebar using the st.sidebar function to allow the user to input the episode ID, which is then followed by clicking the "Submit" button. Once clicked, the application will transcribe and summarize the audio of the episode. After a few minutes, the outcomes will appear on the website. If the user wishes to download the results, a "Download" button is available to compress the transcription and summary into a single file. The system automatically generates both the local URL and the Network URL, allowing the user to select either link to obtain the desired result. As a result, we now have a wonderful app that can transcribe and summarize your favourite podcast!

VI. RESULTS

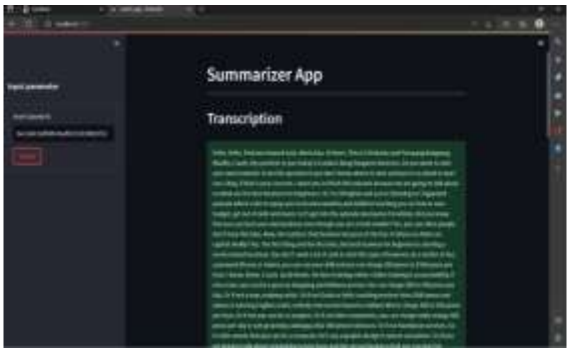

Fig. 2. Result

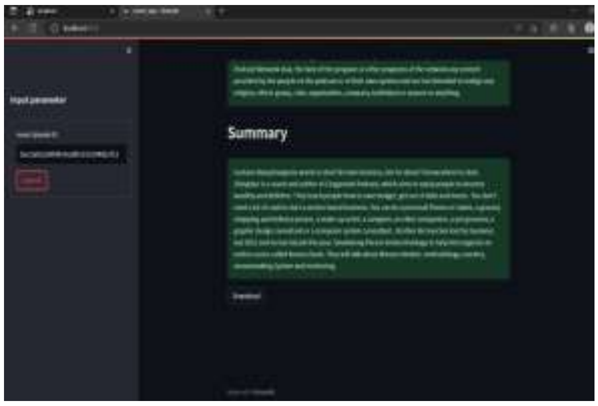

Fig. 3. Result

VII. CONCLUSION

This project aims to demonstrate the proof-of-concept and provide a direction for future development of a fully automated method for summarizing podcast speech. Due to the complexity of this task, there is considerable room for improvement. Podcasts typically require active attention from listeners for extended periods of time, unlike listening to music. The primary input for processing is an audio file containing human speech, which may be recorded live or pre-recorded. However, subjective elements such as the speaker's style, humor, or production quality can be challenging to discern from a text description. The generated summaries contain clear and understandable audio information.

VIII. RECOMMENDATION

While our predictions achieved a high accuracy of 99.6%, it is important to note that no model can be completely perfect. One limitation of this model is that it may not accurately identify species that it has not been trained on, although it will still make a prediction that closely matches the correct type. Nonetheless, this model has potential for further improvement through the collection and analysis of more data, ultimately leading to a real-time audio summarization system.

REFERENCES

[1] A. Vartakavi, A. Garg, and Z. Rafii, "Audio Summarization for Podcasts" 29th European Signal Processing Conference (EUSIPCO), 2021.

[2] Pazhani. A, A. J., Gunasekaran, P., Shanmuganathan, V., Lim, S., Madasamy, K., Manoharan, R., &Verma, A. (2022).Peer–Peer Communication Using Novel Slice Handover Algorithm for 5G Wireless Networks.Journal of Sensor and Actuator Networks, 11(4), 82.

[3] M. H. Su, C. H. Wu, and H. T. Cheng "Two-stage transformer-based approach for variable-length abstractive summarization in the IEEE/ACM Transactions on Audio," Speech, and Language Processing, 2020.

[4] A. Gidiotis and G. Tsoumakas,"A divide-and-conquer method for summarizing long documents," IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2020.

[5] R. K. Yadav, R. Bharti, R. Nagar, and S. Kumar, "A Model For Recapitulating Audio Messages Using Machine Learning," 2020 International Conference for Emerging Technology (INCET).

[6] "Transforming Wikipedia Into Augmented Data for Query-Focused Summarization," IEEE/ACM Transactions on Audio, Speech, and Language Processing in 2022, and discusses a method for using Wikipedia as a source of augmented data for query-focused summarization.

[7] "Analysis of Real Time Video Summarization using Subtitles," 2021 International Conference on Industrial Electronics Research and Applications (ICIERA).

[8] Dhanabalan, S. S., Sitharthan, R., Madurakavi, K., Thirumurugan, A., Rajesh, M., Avaninathan, S. R., & Carrasco, M. F. (2022). Flexible compact system for wearable health monitoring applications.Computers and Electrical Engineering, 102, 108130.