

DIGITAL SUMMARIZATION OF PODCASTS IN REAL TIME

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

G.Sathvik Sri Harsha (Reg.No - 39110349)
Barma Bharath (Reg.No – 39110136)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A” by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119**

APRIL- 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with 'A' grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **G.Sathvik Sri Harsha(39110349)** and **Barma Bharath(39110136)** who carried out Project Phase-2 entitled “**DIGITAL SUMMARIZATION OF PODCASTS IN REAL TIME**” under my supervision from June 2022 to November 2023.

Internal Guide

Ms. R.YOGITHA M.E., Ph.D

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.



Submitted for Viva-voce Examination held on 20.04.2023

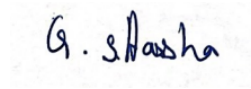
Internal Examiner

External Examiner

DECLARATION

I, **G.Sathvik Sri Harsha(Reg.No- 39110349)** and **Barma Bharath(39110136)**, hereby declare that the Project report” **DIGITAL SUMMARIZATION OF PODCASTS IN REAL TIME**” done by me under the guidance of **Ms. R. Yogitha M.E., Ph.D** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 20.04.2023
PLACE:Chennai

A handwritten signature in blue ink, reading "G. Sathvik Sri Harsha", is placed over a light blue rectangular stamp.

SIGNATURE OF THECANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews .

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. R. Yogitha M.E., Ph.D** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

We offer a one-of-a-kind method for automatically creating podcast audio summaries, allowing listeners to effortlessly preview podcast episodes. The proposed technique transcribes audio from a podcast using automatic speech recognition (ASR), then extracts text summarization to summarize the transcript, and finally provides the audio linked with the text summary. Due to a shortage of relevant datasets for our study, we created our own by transcribing audio from multiple podcasts and creating summaries for these transcripts using a manual annotation tool. Using these text summaries, we fine-tuned a recent Transformer-based summarizing system to precisely handle podcast summaries. ROUGE-(1/2/L) F-scores of 0.63/0.53/0.63 are produced by our method, showing good podcast summarization performance.

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	vii
1	INTRODUCTION	8
2	LITERATURE SURVEY	13
	2.1 Inferences from Literature Survey	19
	2.2 Open problems in Existing System	19
3	REQUIREMENTS ANALYSIS	20
	3.1 Feasibility Studies/Risk Analysis of the Project	20
	3.2 Software Requirements Specification Document	21
4	DESCRIPTION OF PROPOSED SYSTEM	22
	4.1 Selected Methodology or process model	22
	4.2 Architecture / Overall Design of Proposed System	27
	4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System	27
	4.4 Project Management Plan	31
	4.5 Financial report on estimated costing (if applicable)	
	4.6 Transition/ Software to Operations Plan (as applicable)	
	REFERENCES	31

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
4.2.1	Proposed architecture diagram	27

CHAPTER 1

INTRODUCTION

Summarization is the process of compressing facts into a form that is simple to read and understand, as well as the technique of expressing the essence of the information while keeping the grammar and semantics of the language. The method of recognizing text from pictures or digitalized materials is referred to as optical character recognition. When a user uploads a pdf file to our tool, it is first transformed into an image array using the popular pdf2image python function. The pdf2image module is a wrapper over the command line programs pdf2ppm and pdf2cairo for converting PDFs to PIL image lists. The list of photos must be sent to the OCR. Each photograph is digitalized in order to extract text information. A combination of model recognition and characteristic detection is used to recognize the letters, and hence the words, in the picture. A string containing the recognized text is returned.

Because sites like YouTube make it so easy to access enormous amounts of internet content, multimedia summary has become an essential need. In general, automated summarizing seeks to give a shortened and informative version of the original material. This essay focuses on the form of automatic summarization that correlates to an audio stream. There are three approaches to creating an audio summary: controlling the summary using solely audio functions, extracting the text from the audio signal and directing the summarizing process with textual techniques, and a hybrid approach that combines the first two.

Each method has advantages and disadvantages. The advantage of using only audio attributes to construct a summary is that it is fully independent of transcription; however, this can also be a disadvantage because the abstract is based only on how things are stated. In contrast, starting the summary with textual approaches has advantages from the information contained in the text, dealing with more informative summaries; nevertheless, in some cases, transcripts are not available. Finally, the use of audio functions and textual methods can improve the quality of the summary; however, the disadvantages of both approaches exist.

The multidisciplinary field of voice recognition employs specific technology to recognize spoken language and transform it into text. The user's audio file is originally pre-processed for transcription. A Hidden Markov Model (HMM) template was used to create an audio summarizing approach. To describe HMM observational vectors, they employed a variety of acoustic/prosodic characteristics: speech rate; F0 min, max, average, span, and slope; RMS min, max, and average energy; RMS slope and sentence length. The hidden variables indicated whether or not a segment was included or removed from the summary. They employed conventional accuracy, recall, and F-measure information retrieval metrics on more than 20 CNN transmissions and 216 stories previously used in the assessment.

The findings demonstrate that the HMM frame has excellent coverage (Recall = 0.95) but low precision (Precision = 0.26) when picking relevant segments. To recognize speech, most common speech recognition modules employ hidden Markov models. Python's speech recognition module is a package that wraps common voice recognition APIs in an envelope. It returns a string containing the recognized speech. Text synthesis is a prevalent challenge in natural language processing, and there are various proposed techniques that generate consistent and fluid abstractions for the provided text. There are four approaches to summarize text: single document summary, multiple document summary, query-based summary, and informative summary.

The suggested tool successfully achieves a multi-document summary. The textual summary falls into two categories: extractive and abstract textual summaries. The Audio summarization challenge was further addressed by investigating the possibilities of a modulation model for detecting perceptually significant audio occurrences. They calculated the salience of the audio streams using a variety of salience models using linear, adaptive, and non-linear fusion patterns. Experiments were conducted using audio data from six 30-minute sessions. The findings were presented as frame-level accuracy scores, demonstrating that non-linear fusion techniques produce the greatest outcomes.

An audio summary may be created in three ways: by controlling the summary using just audio functions, by extracting the text inside the audio signal and directing the summation process with textual techniques, or by combining the first two. Each method has advantages and disadvantages. The advantage of using only audio attributes to construct a summary is that it is fully independent of transcription; however, this can also be a disadvantage because the abstract is based only on how things are stated. In contrast, using textual approaches to lead the summary benefits from the information provided in the text, resulting in more useful summaries; however, transcripts are not always accessible. Finally, the employment of audio functions and textual methods can increase the quality of the summary; nevertheless, both approaches have drawbacks.

The goal of audio summary without written representation is to create a condensed and informative version of an audio source utilizing just the information available in the audio signal. This type of summary is tough since the information supplied correlates to how things are expressed, which is advantageous in terms of transcript availability. To pick relevant segments and provide an information summary, hybrid audio summarizing approaches or text-based audio summarization algorithms require automatic or manual speech transcripts. Nonetheless, speech transcripts might be expensive, inaccessible, or of low quality, which has an impact on summarization performance.

Indicators are represented by assessing the relevance of a phrase based on its qualities rather than the number of words. Graphic approaches and machine learning models may be employed for this aim. The graphical method is built on top of the page ranking algorithm. The sentences are regarded as nodes in a graph, and an edge is formed between the two nodes representing these phrases based on the measure of similarity between these sentences. They are also weighted based on this similarity metric. As a result, the sentence length is calculated. Audio synthesis based only on acoustic variables such as fundamental frequencies, energy, volume shift, and speaker turn has the significant benefit of not requiring any textual input. This method is especially beneficial when human transcripts for spoken texts are

unavailable and Automatic Speech Recognition (ASR) transcripts have a high word mistake rate. However, in highly informational situations such as televised news, newsletters, or reports, the most significant information is about what is said, whereas audio functions are limited to how things are expressed.

Football is watched by billions of people worldwide, and the amount of content available, including audio and video recordings, is continually increasing. In this scenario, game summaries that combine streams of critical occurrences, such as goals, maps, backups, and penalties, are essential. However, preparing such summaries necessitates time-consuming and laborious event detection and clipping processes, which are frequently performed redundantly by multiple players for different objectives. To recognize emphasized occurrences, audio-based summaries of a soccer match were generated using re-transmissions. Their detection system was built on two acoustic properties: block energy and acoustic repeater indices. Target recall and summary accuracy were assessed, and both categories showed high rates of performance.

Music information research (RIV) is a multidisciplinary subject that focuses on extracting information from music and applying that information to address a broad range of challenges, including automatic song recognition. Because of the great increase of audio distribution channels, such as radio stations, Internet services, download and file exchange services, this application is becoming increasingly significant (including peer-to-peer networks).

The audio summary approach may also be used to summarize audio from recorded meetings. To construct the system, the frame extraction technique is employed initially, which results in the production of text. Following that, the text is retrieved from the photos using natural language processing algorithms. Furthermore, the audio from the meeting is used as input, and a speech recognition template is used to transform it to text. The text generated from those sources is then delivered to the table of contents, which summarizes the information.

The goal of extractive summarization is to create a summary of the content by sequencing the phrases. The efficiency of this method is highly dependent on the

quality of the sentence's features. Most earlier algorithms, on the other hand, required that the characteristics used to represent sentences be built by hand. It was a prerequisite for many previous algorithms. In recent years, the integration of previously separate deep learning approaches has become a major trend. Techniques for integrating previously acquired words have been effectively used. Outstanding performance across a wide range of Natural Language Processing (NLP) initiatives.

The Term Frequency-Inverse Document Frequency (TF-IDF) Summarization Tool suggested in this research summarizes data using the Term Frequency-Inverse Document Frequency method. This utility uses multimedia data that includes text (.txt,.pdf) and audio (.mp3,.wav,.m4a) files. The pdf files are preprocessed with optical character recognition (OCR), and the audio files with the Python Speech Recognition (SR) module. The TF-IDF technique is used for summarization after pre-processing. The programme is intended to do OCR, SR, and text summaries consistently. Users may also save live audio information online and summarize it in real time using the application. The python speech recognition (SR) plug-in is utilized to do this. Finally, a string is returned.

CHAPTER 2

LITERATURE SURVEY

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

We provide a cutting-edge technology solution that will automatically create audio summaries for podcasts. These summaries may be accessed via our website. Because of this functionality, listeners will be able to easily preview podcast episodes before downloading them to their devices. Following the transcription of the audio from a podcast using automated speech recognition (ASR), the transcript was then summarized using extractive text summarization, and ultimately, the audio that is related with the text summary was returned. Using the strategy that was suggested, the phases are carried out in this particular order. Because there were not enough available datasets that were relevant to our endeavour, we came to the realisation that we would need to build our own. We first employed a way of manual annotation to transcribe the audio from a number of podcasts, and then we used the transcribed audio to generate summaries using the manual annotation approach. The end result was that we were successful in achieving this objective. We have only just recently built a model for summarizing that is based on the Transformer, and with the aid of these text summaries, we have fine-tuned it to especially handle podcast summaries. Our method for summarizing podcasts has ROUGE-(1/2/L) F-scores of 0.63, 0.53, and 0.63, respectively, showing that it achieves an outstanding degree of performance.

[2] H. Lee and G. Lee, "Hierarchical Model For Long-Length Video Summarization With Adversarially Enhanced Audio/Visual Features," *2020 IEEE International Conference on Image Processing (ICIP)*, 2020

In this paper, we propose a novel supervised method for summarizing long-length videos. Many recent approaches presented promising results in video summarization. However, videos in most benchmark datasets are short in duration (< 10 minutes), and the methods often do not work well for very long-length videos (>1 hour). Furthermore, most approaches only use visual features, while audios

provide useful information for the task. Based on these observations, we present a model that exploits both audio and visual features. To handle long videos, the hierarchical structure of our model captures both the short-term and long-term temporal dependencies. Our model also refines the extracted features using adversarial networks. To demonstrate our model, we have collected a new dataset of 28 baseball (~3.5 hours) videos, accompanied by an editorial summary video that is 5% in length of the original video. Evaluation on the dataset suggests that our method produces quality summaries for very long videos.

[3] M. -H. Su, C. -H. Wu and H. -T. Cheng, "A Two-Stage Transformer-Based Approach for Variable-Length Abstractive Summarization," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020.

This study proposes a two-stage method for variable-length abstractive summarization. This is an improvement over previous models, in that the proposed approach can simultaneously achieve fluent and variable-length abstractive summarization. The proposed abstractive summarization model consists of a text segmentation module and a two-stage Transformer-based summarization module. First, the text segmentation module utilizes a pre-trained Bidirectional Encoder Representations from Transformers (BERT) and a bidirectional long short-term memory (LSTM) to divide the input text into segments. An extractive model based on the BERT-based summarization model (BERTSUM) is then constructed to extract the most important sentence from each segment. For training the two-stage summarization model, first, the extracted sentences are used to train the document summarization module in the second stage. Next, the segments are used to train the segment summarization module in the first stage by simultaneously considering the outputs of the segment summarization module and the pre-trained second-stage document summarization module. The parameters of the segment summarization module are updated by considering the loss scores of the document summarization module as well as the segment summarization module. Finally, collaborative training is applied to alternately train the segment summarization module and the document summarization module until convergence. For testing, the outputs of the segment summarization module are concatenated to provide the variable-length abstractive summarization result. For evaluation, the BERT-biLSTM-based text segmentation model is evaluated using ChWiki_181k database and obtains a good effect in

capturing the relationship between sentences. Finally, the proposed variable-length abstractive summarization system achieved a maximum of 70.0% accuracy in human subjective evaluation on the LCSTS dataset.

[4]Y. He, X. Xu, X. Liu, W. Ou and H. Lu, "Multimodal Transformer Networks with Latent Interaction for Audio-Visual Event Localization," *2021 IEEE International Conference on Multimedia and Expo (ICME)*, 2021

The task of audio-visual event localization (AVEL) aims to localize a visible and audible event in a video. Previous methods first divide a video into segments and then fuse visual and acoustic features at the segment level via a co-attention mechanism. However, existing methods mostly model relations between individual visual and audio segments in a limitedly short period, which may not cover a longer video duration for better high-level event information modeling. In this paper, we proposed a novel model termed Multimodal Transformer Network with Latent Interaction (MTNLI) to tackle this problem. The proposed MTNLI model employs a multimodal Transformer structure to learn the cross-modality relationships between latent visual and audio summarizations in long segment sequences, which summarize the visual and audio segments into a small number of latent representations to avoid modeling uninformative individual visual-audio relations. The cross-modality information between the latent summarizations is propagated to fuse valuable information from both modalities, which can effectively handle large temporal inconsistent between vision and audio. Our MTNLI method achieves state-of-the-art performance on the benchmark AVE (Audio-Visual Event) dataset for the event localization task.

[5] A. Gidiotis and G. Tsoumakas, "A Divide-and-Conquer Approach to the Summarization of Long Documents," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020

We present a novel divide-and-conquer method for the neural summarization of long documents. Our method exploits the discourse structure of the document and uses sentence similarity to split the problem into an ensemble of smaller summarization problems. In particular, we break a long document and its summary into multiple source-target pairs, which are used for training a model that learns to summarize

each part of the document separately. These partial summaries are then combined in order to produce a final complete summary. With this approach we can decompose the problem of long document summarization into smaller and simpler problems, reducing computational complexity and creating more training examples, which at the same time contain less noise in the target summaries compared to the standard approach. We demonstrate that this approach paired with different summarization models, including sequence-to-sequence RNNs and Transformers, can lead to improved summarization performance. Our best models achieve results that are on par with the state-of-the-art in two publicly available datasets of academic articles.

[6] 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)*, 2020

This model aims to develop an efficient way to recapitulate large audio messages or clips for valuable insights. With increase in utilization of audio/visual data day by day, there is a need to handle audio files more intelligently. In this document, a novel approach is presented to build a summarized audio for a given long audio file. This method is composed primarily of three modules namely: Conversion of Speech into Text, Text summarization, and lastly conversion of text into speech. Each module is fed by the output of another module except speech to text conversion where input is the given audio file for which summary has to be formed. The first step in audio recapitulation is conversion of given audio to text. This is made possible by sending asynchronous requests to Google Cloud speech API. The next module accomplishes its task of extracting important sentences from the transcript by using the Text Rank algorithm. The last module is to convert the summarized text generated from the output of text summarization module to an audio file. This whole method is given a suitable User Interface using flask and thus a web application is formed for helping users to interact with this model.

[7] H. Zhu, L. Dong, F. Wei, B. Qin and T. Liu, "Transforming Wikipedia Into Augmented Data for Query-Focused Summarization," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2022

The difficulty in successfully training data-driven summarization models may be

attributed to the small size of the current query-focused summarizing datasets. In the meanwhile, the manual building of a query-focused summarization corpus is a time-consuming and resource-intensive process. In this article, we make use of Wikipedia to automatically generate a large query-focused summarization dataset consisting of more than 280,000 samples. This dataset, which we call WikiRef, may be utilized as a method of data augmentation. In addition to this, we create a query-focused summarizing model that is based on BERT and we call it Q-BERT. This model is used to extract sentences from the documents and utilise them as summaries. We only identify and fine-tune a sparse subnetwork, which corresponds to a small portion of the complete model's parameters, in order to better adapt a massive model with millions of parameters to tiny benchmarks. This allows us to better fit the massive model to the tiny benchmarks. The results of experiments conducted on three DUC benchmarks indicate that the model that was pre-trained on WikiRef has already attained a level of performance that is considered to be satisfactory. Following optimization on the particular benchmark datasets, the model that includes data augmentation performs better than strong comparison systems. Additionally, the performance of the model is further improved because to our suggested Q-BERT model as well as subnetwork fine-tuning.

[8] P. G. Shambharkar and R. Goel, "Analysis of Real Time Video Summarization using Subtitles," *2021 International Conference on Industrial Electronics Research and Applications (ICIERA)*, 2021

It is becoming more important to be able to successfully watch user-generated videos due to the fast development in the number of videos created by users. By locating and choosing illustrative stills from the film, video summary is seen as a potentially useful approach to making the most of the information contained inside videos. As the quantity of available video information continues to increase at a fast pace, having access to an automated video summary would be beneficial for anybody who values their time and wants to learn more while spending less of it. Using text summarization and video mapping algorithms, this article demonstrates how key aspects of a video may be extracted from the film's subtitles and used to construct a summary of the video's contents. The version of the summary subtitle file that was created with audio will be played together with the video that was summarized. The ultimate output of the system would be a video summary that

would be complemented by an audio version that was derived from the summary.

[9] P. Gupta, S. Nigam and R. Singh, "A Ranking based Language Model for Automatic Extractive Text Summarization," *2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR)*, 2022

Text, audio, and video files constitute a new "world of data" that has emerged as a consequence of the proliferation of the Internet and other forms of social media. When using the available media, it is quite challenging for a user to either receive an accurate overview or grasp the facts that are relevant and vital. On addition, users or assessors of these data files are interested only in the relevant material or summary that can be acquired from the source files in the shortest amount of time. The only method to summarize a single document or numerous documents in order to collect relevant material from the source files is to use automatic text summarizing, also known as ATS. Because of their erroneous encoding, the available ATS systems provide poor summaries and need a significant amount of time and space to process lengthy documents. As a result, within the scope of this study, we have presented a technique for extracting text summarization that makes use of sentence rating. Experiments have been run on BBC and CNN news datasets, and the results have been assessed with respect to ROUGE using the N-gram Language Model. The success of the suggested methodology for use with news datasets is shown by the quantitative values of the metrics.

[10] B. Zhao, M. Gong and X. Li, "Audio Visual Video Summarization," in *IEEE Transactions on Neural Networks and Learning Systems*

There are two primary modalities that make up video data: audio and vision. Recent years have seen a rise in interest in multimodal learning, particularly for audiovisual learning; this is significant since multimodal learning has been shown to improve the performance of a variety of computer vision tasks. On the other hand, the majority of the currently available methods for summarizing videos only use the visual information while ignoring the audio information. In this condensed paper, we claim that the audio modality may help the visual modality better grasp the video's content and structure, which in turn can aid the process of summarizing. As a result of this, we propose to jointly use the audio and visual information for the video summarization challenge and to create an audiovisual recurrent network (AVRN) in

order to accomplish this goal. To be more specific, the proposed AVRN may be broken down into the following three sections: 1) The two-stream long-short term memory (LSTM) is used to encode the audio and visual feature sequentially by capturing their temporal dependence. 2) The audiovisual fusion is utilized to combine the encoded audio and visual features. LSTM is utilized to combine the two modalities by investigating the latent consistency that exists between them; and 3) the self-attention video encoder is employed in order to capture the global dependence that is present in the video. In the end, the merged audiovisual information as well as the integrated temporal and global dependencies are used in conjunction with one another to anticipate the video summary. In a more tangible sense, the experimental findings on the two benchmarks, namely Sum Me and TV sum, have proved the efficacy of each component and the superiority of AVRN in comparison to those systems that merely utilize visual information for the purpose of video summarizing.

2.1 INFERENCES FROM LITREATURE SURVEY

According to the above-mentioned literature, there has been successful study on Audio Summarization in real time, and several models have been developed.

It is evident that the above mentioned systems have their own pros and cons.

While several recent studies use hybrid technologies to improve accuracy, they are still far from what is required.

Higher precision necessitates cheap computing costs, fast processing, and, most importantly, ease of use.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

The existing system only converts voice audio to text format

The existing system uses python speech recognition library which might not work for all the inputs since the library has dependency issues

The existing system cant analyze if a big dataset of audio files are passed as an input because the existing system isn't trained with large dataset.

CHAPTER 3

REQUIREMENT

ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

FEASIBILITY STUDY

In this phase, the viability of the project is increased server performance, and a business proposal is presented with a very generic plan for the project and some cost estimates. The feasibility assessment of the proposed system is to be carried out during system analysis. Understanding the system's primary needs is required for feasibility study. The Three key considerations involved in the feasibility analysis are

- Economical feasibility
- Technical feasibility
- Operational feasibility

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have modest requirements, as only minimal or null changes are required for implementing this system.

OPERATIONAL FEASIBILITY

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user

must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Hardware specifications:

- o Microsoft Server enabled computers, preferably workstations
- o Higher RAM, of about 4GB or above
- o Processor of frequency 1.5GHz or above

Software specifications:

- o Python 3.6 and higher
- o Anaconda software

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

The recent stratospheric surge in popularity of podcasts presents both a big opportunity and an altogether new set of issues for the content search and recommendation systems that are presently in place. Unlike listening to music, listening to a podcast usually necessitates the listener's total attention for lengthy periods of time. Subjective variables such as the speaker's presenting style, the type of humour utilized, or the production quality may impact the listener's taste; yet, determining these features from a written description is challenging.

In the domain of video, movie trailers allow viewers to receive a sneak peek at some of the film's material before making an informed decision about whether or not to watch the film. Because podcast episodes are often broadcast on a regular basis, creating trailers for each new episode would be impractical. The inclusion of audio summaries has been shown to increase the performance of spoken document search engines.

The subject of our most recent technique suggestion is the automatic generation of concise audio summaries of podcasts. These summaries have the ability to educate the listener not just on the podcast's topic matter, but also about its subjective features, such as the presenting style and degree of production quality. When attempting to summarize podcasts, an audio-based summarizing algorithm encounters a unique set of challenges. Podcasts, for example, primarily focus on material delivered through spoken word and commonly incorporate overlapping speech from many speakers, free-form speech, audio effects, background music, and ads.

A supervised learning system that operates in the audio domain would first need to establish the nature of the audio segment in issue in order to accurately evaluate its relevance. This would necessitate manually annotating a large quantity of training data, which is both difficult and time-consuming because it entails listening to the audio in several runs. Despite the fact that podcasts primarily offer material in the form of audible words, summarizing may also be done in the text-domain on the transcript of the episode.

There has been very little research done on automated systems for summarizing podcast audio. The diversity and narrative nature of podcasts, which may contain unscripted speech as well as music, sound effects, and ads, may provide challenges for speech summary algorithms now in use. To address this obstacle, we treat podcast audio summary as a multi-modal data summarization problem. We create an audio description of a podcast using the text-domain as a guide.

As part of this project, we will transcribe and describe one episode of listen note.com, a podcast that tackles a range of data science challenges and gives guidance on how to become a data scientist. The episode will be developed as a web app.

Stream light will be utilized in the application's development, and the Assembly AI API will allow us to transcribe and summarize an episode of our podcast. This application programming interface (API) will first convert the audio to text using automated speech recognition, and then it will produce a summary of the text.

LIBRARIES USED:

1.Streamlit

Stream lit is a free and open-source framework that enables users to swiftly build and share aesthetically pleasing online applications that utilise machine learning and data science. It is a library written in Python that was developed with the machine learning engineers' needs in mind. Data scientists and machine learning engineers are not web developers, and they have no desire to spend weeks learning how to use these frameworks in order to develop online applications. Instead, what they are looking for is a tool that is simpler to use and work with, as long as it can display data and collect

the characteristics that are necessary for modelling. Using Streamlit, you may design an application that has a great appearance while writing only a few lines of code.

Requests

Python users who want to make HTTP requests should use the requests library because it is the industry standard. It hides the complexity of making requests behind an elegant and straightforward API, allowing you to focus instead on interacting with services and consuming data within your application rather than dealing with those complications. Additionally, it enables the sending of additional information to a web server via parameters and headers, as well as the encoding of the server's answers, the detection of problems, and the management of redirects.

The Requests library provides several complex functionality, such as handling HTTP exceptions and authentication, in addition to simplifying the way in which we work with the HTTP operations. These advanced features include:

Zipfile

The format for ZIP files is widely used as a standard for archiving and compression. A ZIP file can be created, read, written to, appended to, and listed using the tools that are provided by this module. Understanding the format, as described in the PKZIP Application Note, is necessary for making more advanced use of this module.

At the moment, this module is unable to deal with ZIP archives that contain several discs. It is capable of working with ZIP files that make use of the ZIP64 extension (that is ZIP files that are more than 4 GiB in size). It is possible to decrypt files that are encrypted and stored in ZIP archives, but it is not possible to produce encrypted files at this time. Due to the fact that the decryption process is implemented in native Python rather than C, it is painfully sluggish.

Json

The JavaScript Object Notation, abbreviated as JSON, is a standardized format for the representation of structured data that is based on the syntax of JavaScript objects. It is widely utilized for the purpose of data transmission in online applications (e.g., sending some data from the server to the client, so it can be displayed on a web page, or vice versa).

Douglas Crockford is credited for popularizing the JSON data format, which is a text-based data format that follows the JavaScript object syntax. JSON can be read (parsed) and generated in many different programming environments, despite the fact that its syntax is strikingly similar to that of the JavaScript object literal. This is despite the fact that JSON can be used independently of JavaScript.

JSON may be represented as a string, which is convenient when you need to send data from one computer to another over a network. When you wish to access the data, you have to first convert it to a native JavaScript object. This is not a significant problem because JavaScript offers a global JSON object with methods that can transform data between the two formats.

Module 1: Getting the PODCAST from the listennotes api

We will create a method that will get the podcast data from the podcast id given as user input.

Extract Episode's URL from Listen Notes

Listen Notes is a robust online search engine and database for podcasts. It offers an API to gain access to podcast data and makes it possible to create new services and applications that are based on it. During the course of this section, we are going to

extract the episode's url from our target podcast by making use of the Listen Notes API.

- To begin, we will have to register for an account in order to retrieve the data and sign up for the free plan in order to make use of the Listen Notes API. With this plan, you have the ability to have a maximum of 300 requests processed each month, which may be sufficient for working on personal projects.
- The next step is to navigate to the Listen Notes page for the podcast, click the episode in which we are interested, and then select the option to "Use API to get this episode."
- Following that, we will be able to modify the language code to Python and select requests from the list of available options so that we can utilize the library later.
- Make sure to insert the code into your notebook or script after you've copied it.

The retrieval of information from the Listen Notes API is being accomplished through the use of a tactical operation known as submitting a GET request to the endpoint of the Listen Notes Podcast API. After everything is said and done, we store the final result as a JSON object. This object contains the episode's URL, which will be required at a later time.

In addition to that, we are importing a JSON file with the name `secrets.json`. This file is quite similar to a dictionary in that it contains a collection of key-value pairs much like a dictionary does. It is necessary for it to hold the API keys for AssemblyAI and Listen Notes, respectively. You will need to log into your accounts to access them.

Listennotes API

Listen Note API is where we can search and get data related to API using podcast id.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

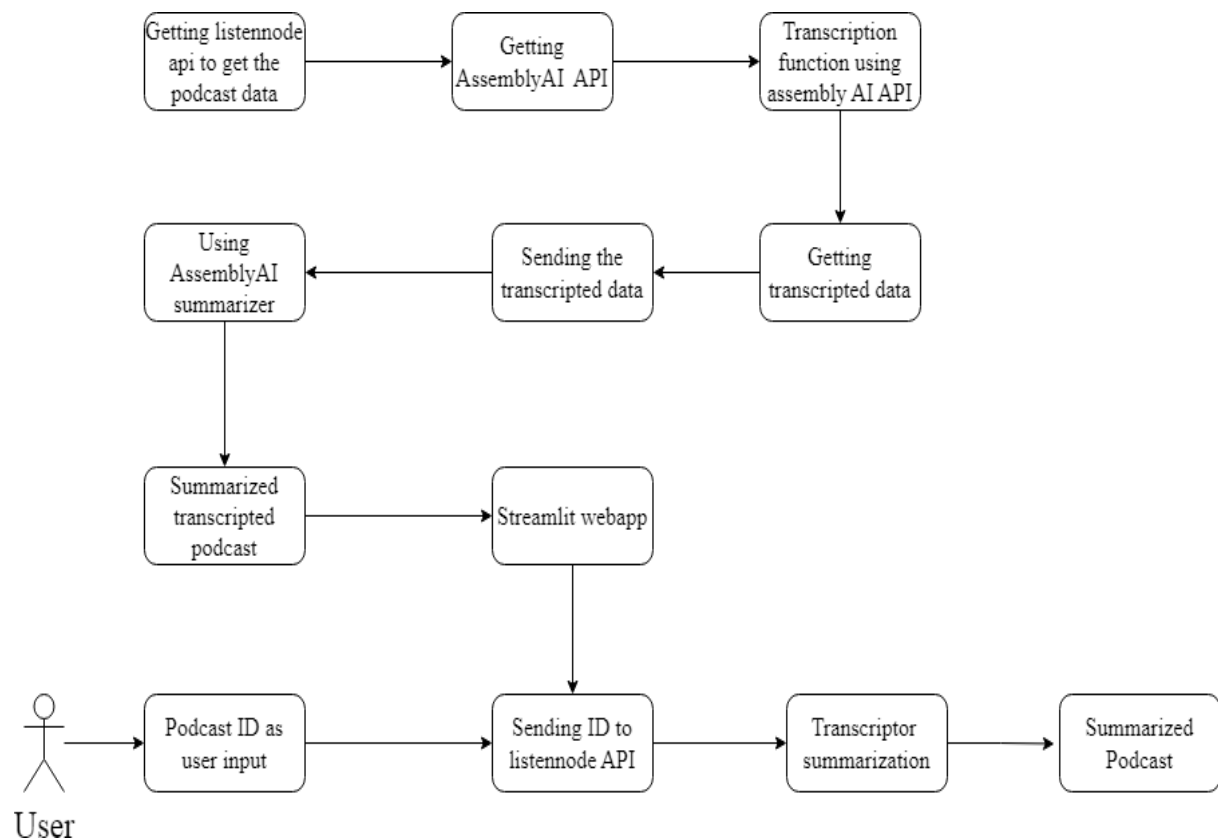


Fig 4.2.1 System Architecture

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

Anaconda is an open-source package manager for Python and R. It is the most popular platform among data science professionals for running Python and R implementations. There are over 300 libraries in data science, so having a robust distribution system for them is a must for any professional in this field. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms. With Anaconda, you can easily set up, manage, and share Conda environments. Moreover, you can deploy any required project with a few clicks when you're using Anaconda. There are many advantages to using Anaconda and the following are the most prominent ones among them: Anaconda is free and open-source. This means you can use it without spending any money. In the data science sector, Anaconda is an industry staple. It is open-source too, which has made it widely popular. If you want to become a data science professional, you must know how to use Anaconda for Python because every recruiter expects you to have this skill. It is a must-have for data science.

It has more than 1500 Python and R data science packages, so you don't face any compatibility issues while collaborating with others. For example, suppose your colleague sends you a project which requires packages called A and B but you only have package A. Without having package B, you wouldn't be able to run the project. Anaconda mitigates the chances of such errors. You can easily collaborate on projects without worrying about any compatibility issues. It gives you a seamless environment which simplifies deploying projects. You can deploy any project with just a few clicks and commands while managing the rest. Anaconda has a thriving community of data scientists and machine learning professionals who use it regularly. If you encounter an issue, chances are, the community has already answered the same. On the other hand, you can also ask people in the community about the issues you face there, it's a very helpful community ready to help new learners. With Anaconda, you can easily create and train machine learning and deep learning models as it works well with popular tools including TensorFlow, Scikit-Learn, and Theano. You can create visualizations by using Bokeh, Holoviews, Matplotlib, and Datashader while using Anaconda.

How to Use Anaconda for Python

Now that we have discussed all the basics in our Python Anaconda tutorial, let's

discuss some fundamental commands you can use to start using this package manager.

Listing All Environments

To begin using Anaconda, you'd need to see how many Conda environments are present in your machine.

```
conda env list
```

It will list all the available Conda environments in your machine.

Creating a New Environment

You can create a new Conda environment by going to the required directory and use this command:

```
conda create -n <your_environment_name>
```

You can replace <your_environment_name> with the name of your environment. After entering this command, conda will ask you if you want to proceed to which you should reply with y:

```
proceed ([y])/n)?
```

On the other hand, if you want to create an environment with a particular version of Python, you should use the following command:

```
conda create -n <your_environment_name> python=3.6
```

Similarly, if you want to create an environment with a particular package, you can use the following command:

```
conda create -n <your_environment_name> pack_name
```

Here, you can replace pack_name with the name of the package you want to use.

If you have a .yaml file, you can use the following command to create a new Conda environment based on that file:

```
conda env create -n <your_environment_name> -f <file_name>.yaml
```

We have also discussed how you can export an existing Conda environment to a .yaml file later in this article.

Activating an Environment

You can activate a Conda environment by using the following command:

```
conda activate <environment_name>
```

You should activate the environment before you start working on the same. Also, replace the term <environment_name> with the environment name you want to activate. On the other hand, if you want to deactivate an environment use the

following command:

```
conda deactivate
```

Installing Packages in an Environment

Now that you have an activated environment, you can install packages into it by using the following command:

```
conda install <pack_name>
```

Replace the term <pack_name> with the name of the package you want to install in your Conda environment while using this command.

Updating Packages in an Environment

If you want to update the packages present in a particular Conda environment, you should use the following command:

```
conda update
```

The above command will update all the packages present in the environment. However, if you want to update a package to a certain version, you will need to use the following command:

```
conda install <package_name>=<version>
```

Exporting an Environment Configuration

Suppose you want to share your project with someone else (colleague, friend, etc.). While you can share the directory on Github, it would have many Python packages, making the transfer process very challenging. Instead of that, you can create an environment configuration .yaml file and share it with that person. Now, they can create an environment like your one by using the .yaml file.

For exporting the environment to the .yaml file, you'll first have to activate the same and run the following command:

```
conda env export ><file_name>.yaml
```

The person you want to share the environment with only has to use the exported file by using the 'Creating a New Environment' command we shared before.

Removing a Package from an Environment

If you want to uninstall a package from a specific Conda environment, use the following command:

```
conda remove -n <env_name><package_name>
```

On the other hand, if you want to uninstall a package from an activated environment, you'd have to use the following command:

```
conda remove <package_name>
```

Deleting an Environment

Sometimes, you don't need to add a new environment but remove one. In such cases, you must know how to delete a Conda environment, which you can do so by using the following command:

```
conda env remove --name <env_name>
```

The above command would delete the Conda environment right away.

4.4 PROJECT MANAGEMENT PLAN

Introduction:	September 1-30
Literature Survey:	October 1-31
System Design:	November 1-30
System Implementation:	December 1-31
Testing:	January 1-30

REFERENCES:-

[1] A. Vartakavi, A. Garg and Z. Rafii, "Audio Summarization for Podcasts," 2021 29th

European Signal Processing Conference (EUSIPCO), 2021

- [2] H. Lee and G. Lee, "Hierarchical Model For Long-Length Video Summarization With Adversarially Enhanced Audio/Visual Features," *2020 IEEE International Conference on Image Processing (ICIP)*, 2020
- [3] M. -H. Su, C. -H. Wu and H. -T. Cheng, "A Two-Stage Transformer-Based Approach for Variable-Length Abstractive Summarization," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020.
- [4] Y. He, X. Xu, X. Liu, W. Ou and H. Lu, "Multimodal Transformer Networks with Latent Interaction for Audio-Visual Event Localization," *2021 IEEE International Conference on Multimedia and Expo (ICME)*, 2021
- [5] A. Gidiotis and G. Tsoumakas, "A Divide-and-Conquer Approach to the Summarization of Long Documents," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2020
- [6] 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)*, 2020
- [7] H. Zhu, L. Dong, F. Wei, B. Qin and T. Liu, "Transforming Wikipedia Into Augmented Data for Query-Focused Summarization," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2022
- [8] P. G. Shambharkar and R. Goel, "Analysis of Real Time Video Summarization using Subtitles," *2021 International Conference on Industrial Electronics Research and Applications (ICIERA)*, 2021
- [9] P. Gupta, S. Nigam and R. Singh, "A Ranking based Language Model for Automatic Extractive Text Summarization," *2022 First International Conference on Artificial Intelligence Trends and Pattern Recognition (ICAITPR)*, 2022
- [10] B. Zhao, M. Gong and X. Li, "AudioVisual Video Summarization," in *IEEE Transactions on Neural Networks and Lear*

