Business Meeting Summary Generation

Mr. Sai Prasad Kashi
HOD of CSE(AIML)
MLR Institute of Technology
Telangana, India
saiprasad.kashi@mlrinstitutions.ac.in

Arjula Dheeraj Reddy
MLR Institute of Technology
Telangana, India
arjuladheerajreddy@gmail.com
20r21a6602@mlrinstitutions.ac.in

Anubrolu Chinmaye
MLR Institute of Technology
Telangana, India
chinmaye06@gmail.com
20r21a6601@mlrinstitutions.ac.in

Donthireddy Saketh Reddy
MLR Institute of Technology
Telangana, India
sakethreddydonthireddy@gmail.com
20r21a6614@mlrinstitutions.ac.in

Mohammed Irfan
MLR Institute of Technology
Telangana, India
irfanirfan7273@gmail.com
20r21a6638@mlrinstitutions.ac.in

ABSTRACT

In the dynamic corporate landscape business meetings serve as pivotal platforms for decision-making, teamwork and progress evaluation. This article explores the transformative potential of meeting summarizers emphasizing the need for concise and coherent summaries in the face of time constraints. A novel Natural Language Processing solution is presented to address this imperative requirement. The study critically examines various summarization approaches tracing the evolution from the widely adopted Extractive Summarization to the emerging paradigm of Abstractive Summarization. The research provides in-depth descriptions and comparisons of these methods using comprehensive tables, offering practitioners valuable insights for informed decision-making. Key techniques such as Text Rank Algorithm, TF-IDF, LSTM and K-Means Clustering are scrutinized, contributing to a nuanced understanding of the diverse landscape of meeting summarization. This work aims to advance corporate efficiency by providing a roadmap for the selection and implementation of optimal summarization strategies.

CCS CONCEPTS
Computing methodologies → NLP → Neural Networks
Information systems → Data Generation

Keywords:
Text Summarization, NLP, Sentiment Analysis, Text Rank Algorithm, TF-IDF, LSTM, LSA, T5.

1 INTRODUCTION

In the rapidly evolving landscape of contemporary business, effective communication and efficient information dissemination are critical determinants of success. Business meetings play an indispensable role as key platforms for collaboration, decision-making, and strategy formulation. However, the substantial volume of information exchanged in these sessions poses a challenge for professionals seeking to distill and retain key insights. Recognizing the pressing need to address this challenge, the field of business meeting summary generation has emerged as a focal point for research and development. Traditional note-taking techniques have proven less effective, especially in lengthy meetings where capturing essential details is arduous.

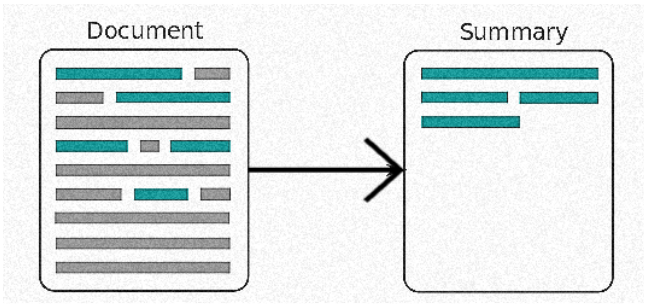


Figure 1: Original Transcript Vs Summary Transcript

1.1 Brief Introduction to NLP

In the realm of contemporary information processing, Natural Language Processing (NLP) emerges as a transformative field, wielding the power to unlock the intricacies of human language. As showcased in the project's exploration of meeting summarization methodologies, NLP plays a central role in deciphering the character-wise understanding of sentence structures. By delving into concepts like syntactic analysis, semantic analysis [2][6] and lexical relations, NLP enables a comprehensive comprehension of language nuances. This is particularly crucial in the context of business meeting summarizations, where understanding the subtle variations in speaking styles and semantic nuances is paramount.

NLP's significance extends beyond mere linguistic analysis; it is an indispensable tool in the realm of summarization. NLP techniques are pivotal in processes such as pre-processing, where the removal of superfluous elements facilitates cleaner analysis and improved transcript processing.

1.2 Introduction to Extractive Summarization

Extractive summarization is a text summarization technique that involves selecting and combining important passages or sentences from the original text to create a summary. Extractive summarization [13] directly uses existing text fragments from the source document. The system first analyzes the input text and identifies sentences that are considered important or representative of the main ideas. This can be based on various criteria such as keyword frequency, sentence length, and semantic similarity. Each sentence is assigned a score based on its importance, relevance, or other criteria. These scores are used to rank the sentences in order of importance. Finally, the system selects a subset of the highest-scoring sentences [26] to include in the summary. The selected sentences are then combined to form the final summary. Extractive summarization [15] is relatively simpler to implement compared to abstractive summarization because it does not involve generating new text. However, it can sometimes produce summaries that are less coherent or fluent, as it relies on selecting and stitching together existing text fragments.

1.3 TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) [10] [16] is one of the most popular techniques in NLP, which turns text transcripts into numerical vectors. It is applied before machine learning algorithms as a type of pre-processing step to textual data.

$$\mathrm{TF}(n,t) = \frac{\text{Frequency of term } n \text{ in transcript } t}{\text{no. of terms in transcript } t}$$

$$\mathrm{IDF}(n,D) = \log\left(\frac{\text{no. of transcripts encompassing term } n}{\text{Total no. of transcripts in the dataset } D}\right)$$

$$\mathrm{TF\text{-}IDF}(n,t,D) = \mathrm{TF}(n,t) \times \mathrm{IDF}(n,D)$$

TF-IDF [18] vectorization is helpful in applications related to information retrieval and text classification tasks, as it captures the importance of sentences in individual transcripts according to their frequency scores in the whole transcript.

1.4 Introduction to Abstractive Summarization

Abstractive summarization [1] is a methodology used under NLP to create succinct summaries that contain the most contextually relevant information from a document. Unlike extractive summarization, abstractive summarization [17] under proper training with neural networks will be able to generate a new set of sentences that cover the key points and important decisions from the longer texts, which in turn is often more challenging and complex to develop. The system analyzes the input text to understand its content, identifying key entities, relationships, and main ideas. Based on this understanding, the system forms a high-level conceptual representation of the text, capturing the main ideas and key information. Using the conceptual representation, the system generates a summary that conveys the essential information in a concise and coherent manner. In some cases, the system may need to combine information from different parts of the text to create a coherent summary. The process involves understanding the sentences in the textual document with the help of semantic, syntactic, and lexical analysis, as well as phonology, which always plays a major role in distinguishing similar sentences and nuances. Abstractive summarization [22] is a challenging task leading to higher computational cycles to develop and train, but the end result is much deserving of the hard work that is required to create concise summaries that are more relevant and contextually concise.

1.5 Role of RNN-LSTMS in Summarization

Recurrent Neural Networks (RNNs) [3] are a subsidiary of neural networks that work with sequential data, therefore making them most desirable for tasks like NLP and text classification. Due to the limitations of RNN's ability to work under extreme conditions like long-term dependencies, LSTMs are created that are able to mitigate these limitations. RNNs and LSTMs [20] have a similar work flow, i.e., processing sequential data one after another by creating a hidden state, which is helpful to achieve the objective. Even Though RNN and LSTM are similar in nature, LSTMs have a more comprehensive architecture, which is seen in the arrangement of memory cells to access and store information for longer dependencies. Generally, LSTMs have three major gates: the input gate, the output gate, and the forget gate. The input gate decides whether or not to store new sequential data in the memory cell. The output gate decides the output of the LSTM cell based on current input data and the newly updated memory cell. The forget gate decides which information is to be stored or discarded according to the previous hidden state and current input state. LSTMs are trained using backpropagation [14] through time (BPTT), which is one of the variant of backpropagation that takes into account about the sequential nature of the data. LSTMs are often used in various Natural Language Processing tasks, including text generation, language modeling and machine translation, because of its ability to capture long-range dependencies in sequential data.

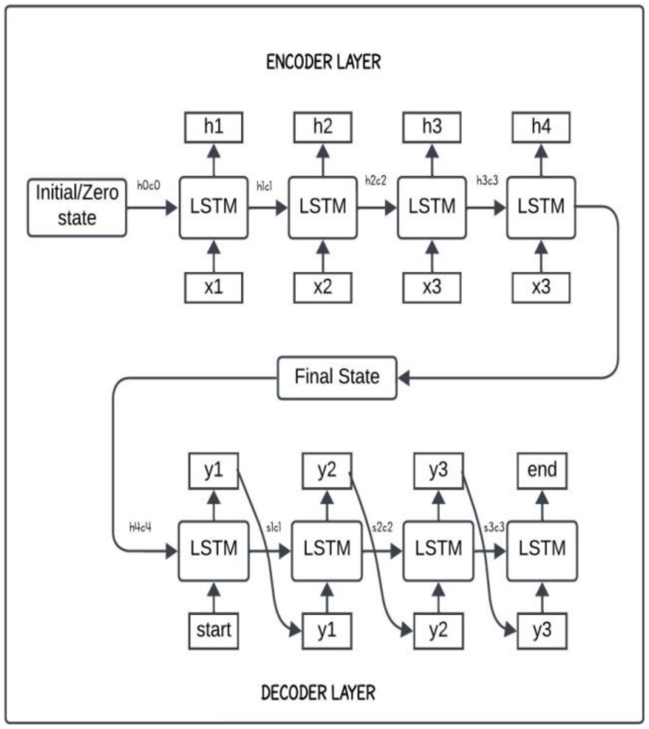


Figure 2: Encoder Decoder Architecture

2 EXISTING PROBLEMS

There are several challenges that can hinder the effectiveness of a working meeting summarizer. Often, generated summaries might miss important points discussed during the meeting, leading to issues of relevance and accuracy that cannot be neglected. It is essential to ensure that the generated summaries capture the context of the discussions perfectly. Some of the existing systems may struggle to comprehend nuanced contexts, leading to summaries that lack depth or relevance. Many businesses incorporate multimodal data, such as text, audio, and visual content. Existing summarization providers primarily focus on text-based data and do not leverage other modalities. While extractive-based summarization methods extract key sentences from the meeting transcript, there are fewer providers who leverage abstractive methodologies to generate summaries in a more human-like manner. Improving productivity will benefit from the development of scalable summarization algorithms that can handle massive datasets in real-time.

3 PROPOSED SYSTEM

Combining extractive and abstractive summarization techniques. Unlike the extractive method, the abstractive-based technique facilitates the creation of new phrases. The Encoder-Decoder [21] layered system was employed after the succession of processes such as TF-IDF [7] and Text Rank [13] to construct the top-ranked sequences existing in the input transcripts. LSTM [27] memory cells are excellent at capturing long-term relationships in the data. We experimented with training and testing ratios to find the ideal fit where the model gives succinct outputs.

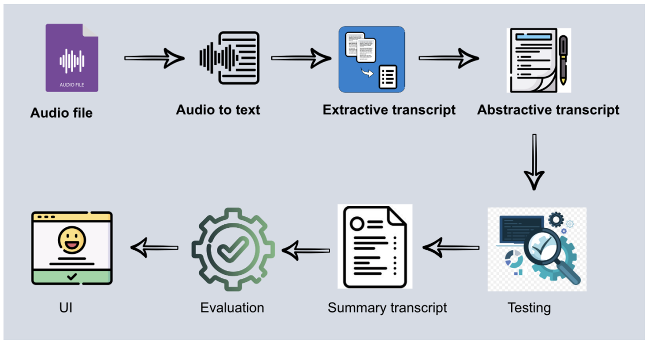


Figure 3: Proposed Architecture

4 IMPLEMENTATION

Our proposed system is a well-structured and defined process following mainly 3 major steps and solves the existing problems and provides an interface to generate concise and coherent summaries.

4.1 Overview

The proposed solution revolves around three main sub-process. They are:
1. Audio-to-Text Transcription
2. Extractive Summarization
3. Abstractive Summarization

4.2 Audio-to-Text Transcription

Our system's initial step is Audio-to-Text Conversion. There are several modules and methods available for this process, including VOSK, IBM Watson, Whisper,ASR, [8] Rev AI, STT system [23] and many more. After a great deal of trial and error and testing all the modules, Whisper has the highest transcription accuracy rate. OpenAI developed Whisper, a potent AI model intended for automated voice recognition (ASR). It takes on the challenge of translating spoken words from audio files into written language. Because of its adaptability, it's a useful tool for transcription of lectures, interviews, meetings, and other situations when audio has to be turned into text.

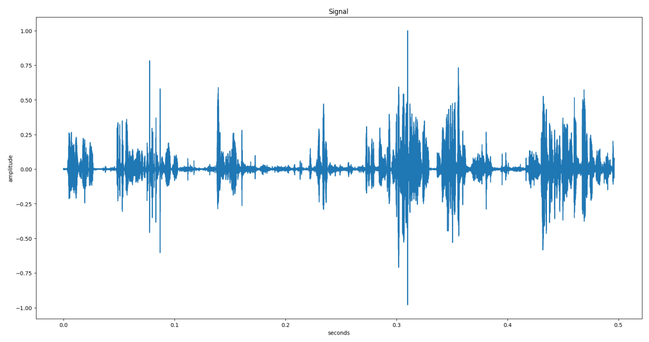


Figure 4: Amplitude vs. Time Representation of the Audio Signal

4.3 Extractive Summarization

The extractive summary step begins after the first phase is finished, using a text transcript captured from the audio file as the input. It passes through several stages given below.

(1) **Cleaning and preparing the text:-** We may take care of punctuation and stop words by using spacy libraries. It eliminates punctuation and stop words from the transcript, which are frequent terms like "the" and "and."
(2) **Calculating Web Frequency:-** We keep track of terms and their frequencies in a dictionary and add to the count whenever a word appears more than once.
(3) **Normalizing Word Scores:-** We calculate the importance score for each word by first determining the highest word frequency in the transcript. We then normalize the word frequency by dividing it by the maximum frequency.
(4) **Sentence Scoring:-** phrase scoring [24] involves creating a dictionary to hold the scores of the sentences. Then, we iterate through the transcript, checking each phrase to see if each word has a normalized frequency score. If it does, we increment the score.
(5) **Selecting Top Sentences:-** [11]We set a starting percentage barrier, say 0.3. We locate and choose the sentences from the sentence score dictionary that have the highest scores using the nlargest function.

(6) **Summary Creation:-** To create a summary, the list of phrases with the highest scores is combined into a single string.

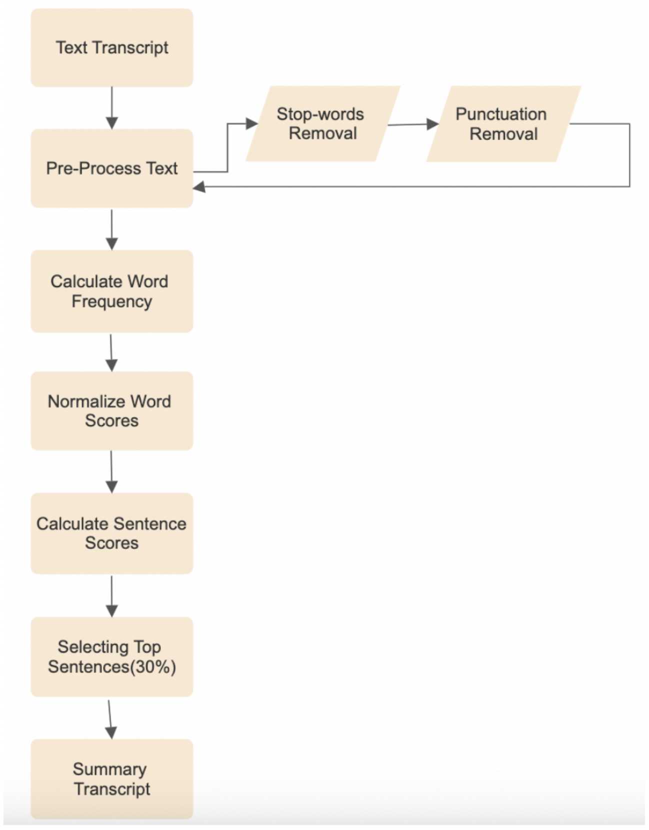


Figure 5: Flowchart of Extractive Summarization

4.4 Abstractive Summarization

Implementing abstractive summarization with the Hugging Face Transformers library on the T5[4] model. To overcome computational constraints, the approach takes a lengthy input text and divides it into manageable parts. It then creates summaries for each chunk separately. The T5 tokenizer and model are loaded from the "t5-small" pre-trained model [9] within the function. The input text is divided into pieces, with chunk length defining the maximum length of each chunk. The tokenizer passes the text to the model to create a summary for each chunk, encoding it with a summarizing prefix ("summarize:"). The tokenizer is used to decode the created summary, and then all of the summaries from the various chunks are combined to create the final summary.

To get the headline or theme of the meeting, train and infer a summarization model using the SimpleT5 library. Initially, the dataset is loaded from a CSV file, which contains headlines and corresponding text. The dataset is then preprocessed to rename columns and add a task-specific prefix ("summarize:") to the source text, as required by T5[12] models. After splitting the dataset into training and testing sets, the SimpleT5 model is initialized and trained on a subset of the data. The training process involves specifying parameters such as maximum token lengths, batch size, and number of epochs. After training, the trained model is saved in the outputs folder. Subsequently, the trained model is loaded for inference, and a sample text is provided for summarization. The model predicts a summary for the given text.

5 SOLUTION TO THE PROBLEM

The below figure describes the flow of the entire proposed solution, including all the necessary modules.

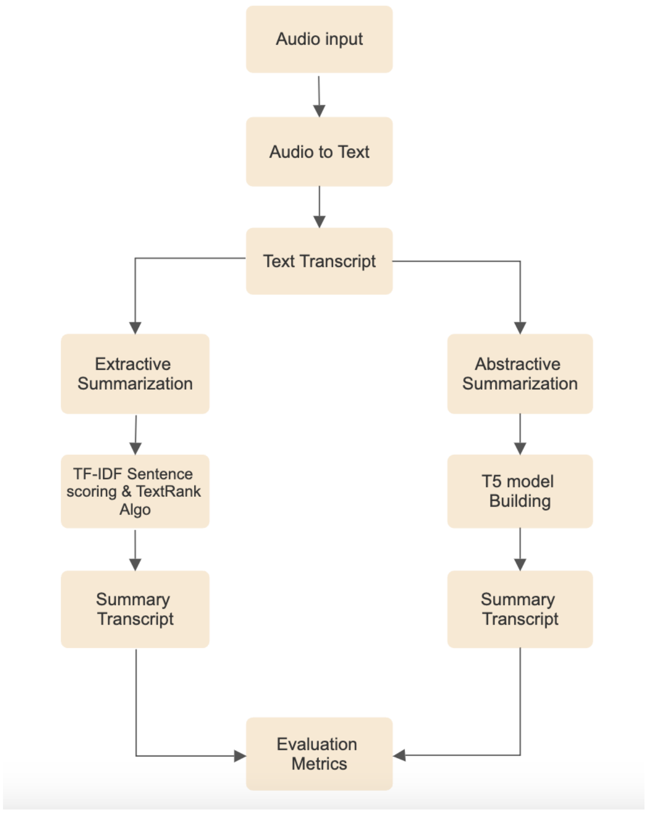


Figure 6: Flowchart of the Proposed Solution

5.1 Integrating Frontend with Backend Using FLASK

Here this pseudocode explains a Flask application that integrates frontend with the backend. From the service page where the input(mp3) file is uploaded and redirected according to their defined path variables. Data is gathered, and different functions help in providing different summaries also generating a theme for the meeting according to its contextual similarity.

```
# Import Flask and other libraries
# Configure the Flask app
# Define the allowed file extensions
ALLOWED_EXTENSIONS = {'mp3'}
# Function to check if a file has an
```

```
allowed extension
# Define the Flask application
app = Flask(__name__, folder="templates")
# Define the main route
@app.route('/')
def index():
    # Render the index.html page  template
# Route to handle audio to text conversion
@app.route('/aud2text',methods=['POST',GET'])
def aud2text():
    if request.method == 'POST':
        #check file authentication
        # Call process_audio function
        # Insert data into a text file
        # Redirect to the aud2text template
# Route to handle audio to summary conversion
@app.route('/aud2sum',method=['POST','GET'])
def aud2sum():
    if request.method == 'POST':
        #check file authentication
        # Call audio_process function
        # Call Ext_Summary function
        # Insert data to a text file
        # Redirect to aud2sum template
# Route to handle headline extraction
@app.route('/headline',method=['POST','GET'])
def headline():
    if request.method == 'POST':
        #check file authentication
        # Call process_audio func
        # Call Abstractive_Summary function
        # Call abs_summ func[trained-model]
        # Redirect to headline page template
# Run the Flask application
if __name__ == '__main__':
    app.run(debug=True)
```

5.2 Speaker Diarization

This pseudocode describes the application that uses the pyannote speaker-diarization[19][5] pipeline to distinguish and identify different speakers speaking in the audio file. The process involves segmenting audio into speaker turns and exporting them to a wav file where it makes use of whisper library for transcription. Overall, the code orchestrates the entire process of speaker diarization, audio segmentation, caption generation, and retrieval.

```
# Import required libraries
# Load Whisper base module
def perform_speaker_diarization():
    # Load the pre-trained pipeline
    # Perform speaker diarization
    # Write diarization res to a text file
    # Func to convert time string to ms.
    # def a audio seg of a certain length.
    # Initialize variables for storing
    segmented audio and segment timestamps.
    # Read the diaz res from the text file.
    for each line in dia.reslts():
        # Extract start and end timestamps.
        # Convert timestamps to milliseconds.
        # Append the start timestamp to the
        list of segment timestamps.
        # Extract the audio segment and
        append it to the segmented audio.
        # Append a silent segment after
        each audio segment.
    # Export the segmented audio to WAV file.
    # Use an ('whisper') to generate captions.
    # Read captions from the new VTT file.
    # Convert caption timestamps to ms and
    store them with the caption text.
    # Add speaker labels to captions
    # Def a list to store speaker-labels
    # Def spacing b/w captions(spacermilli)
    for each segment index:
        # Find the captions for each segment.
        for each caption in the segment:
            # Cal new start-time for caption
            # Determine the speaker label
            based on the segment index
            # Append the formatted caption
            with speaker label to the
            speaker_labels list

    # Return the captions with speaker labels
while True:
    # Take user input
    path = input()
    # load our model
    ans = perform_speaker_diarization(path)
```

5.3 Audio to Text Translation

This pseudocode describes the application that uses the Whisper[25] library to convert the audio file into text format. A base module is utilized in this process. We remove time stamps, IDs, and other information from the output that is produced so that it is only text that we need for our objectives.

```
# Import required libraries
# Load Whisper base module
def process_audio():
    model = whisper.load_model("base")
    result = model.transcribe(filepath)
    res = result["text"]
    return res
while True:
    # Take user input
    filepath = input()
    # load our model
    result = process_audio(filepath)
```

5.4 Text to Text Summary

This pseudocode explains a program that receives an input text, tokenizes it, determines word and sentence scores[24], normalizes the scores, and then creates a summary transcript by combining the best 30 phrases[11].

```
# Import required libraries and modules
# load spacy model
def preprocess(text):
        #intialize stopwords
        #intialize punctuations
        remove stop-words and punctuations
def ext_summary(text):
    text = preprocess(text)
    # load spacy model on text
    # create a empty dictionary
    # iterating the list to get
    word frequency dictionary
    max_freq = max(word_freq.values())
    # normalize the values
    # create a empty sentence_freq dictionary
    # iterate over the input to get
    Sentence scores
    Using TF-IDF and confine similarity
    Initialize a threshold = 0.3
    Select sentences according to threshold
    summary = Combine sentences
    summary = ''.join(summary)
    return summary
while True:
    # Take user input
    input_text = input()
    # load our function
    result = ext_summary(input_text)
```

5.5 Abstractive Summary

This pseudocode explains a program that generates a summary and headline of provided input text using a trained model and pre-trained T5[12] model and tokenizer from the Hugging Face transformers library. It includes routines for preprocessing input text, setting up pad sequences, chunk length and beams to adjust for longer text transcripts.

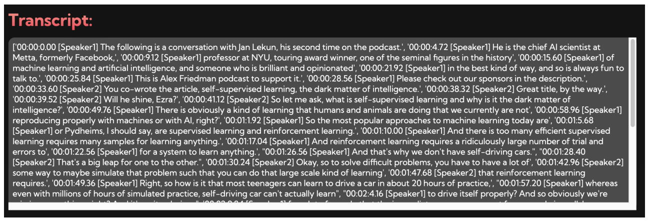
```
# Import required libraries and modules
# load simplet5 trained model
def abs_summary(text):
    # load_model
    # initialize tokenizer with t5tokenizer
    using pertrained model_name
    # split text length into small chunks
    summaries=[]
    for each chunk in chunks:
       prepend "summarize:" to chunk and
       limit the tokens
       # generating a summary using model
       Decode the generated summary and
       store them
   # Join all summaries into a string
    return summary
def theme(text):
    prepend "summarize:" to text
    and store it
    # initialize model as a Simplet5
    # Load a pre-trained T5 model
    named t5-base into model.
    # Load the fine-tuned model
    located at outputs folder
    # predict the result using
    loaded model & store in result

    return result
while True:
    # Take user input
    input_text = input()
    # load our function
    summary = abs_summary(input_text)
    result = theme(input_text)
```

6 RESULTS

This section represents the outputs generated after implementing the proposed system under the different methodologies defined.

6.1 Audio Transcription

This sub-section showcases the output/text generated under speaker-diarization and whisper processes, where the file is divided using the speaker-diarization pipeline to audio segment parts of the file to identify the sentences spoken by a certain speaker, then follows the whisper pre-trained base model to transcribe the individual segments to provide text spoken by a certain individual.



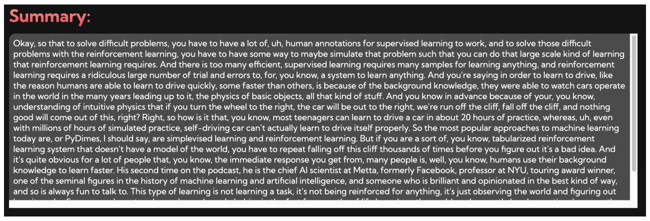
Transcript:

['00:00:0.00 [Speaker1] The following is a conversation with Jan Lekun, his second time on the podcast.', '00:00:4.72 [Speaker1] He is the chief AI scientist at Metta, formerly Facebook,', '00:00:9.12 [Speaker1] professor at NYU, touring award winner, one of the seminal figures in the history', '00:00:15.60 [Speaker1] of machine learning and artificial intelligence, and someone who is brilliant and opinionated', '00:00:21.92 [Speaker1] in the best kind of way, and so is always fun to talk to.', '00:00:25.84 [Speaker1] This is Alex Friedman podcast to support it.', '00:00:28.56 [Speaker1] Please check out our sponsors in the description.', '00:00:33.60 [Speaker2] You co-wrote the article, self-supervised learning, the dark matter of intelligence.', '00:00:38.32 [Speaker2] Great title, by the way.', '00:00:39.52 [Speaker2] Will he shine, Ezra?', '00:00:41.12 [Speaker2] So let me ask, what is self-supervised learning and why is it the dark matter of intelligence?', '00:00:49.76 [Speaker1] There is obviously a kind of learning that humans and animals are doing that we currently are not', '00:00:58.96 [Speaker1] reproducing properly with machines or with AI, right?', '00:01:1.92 [Speaker1] So the most popular approaches to machine learning today are', '00:01:5.68 [Speaker1] or Pydheims, I should say, are supervised learning and reinforcement learning.', '00:01:10.00 [Speaker1] And there is too many efficient supervised learning requires many samples for learning anything.', '00:01:17.04 [Speaker1] And reinforcement learning requires a ridiculously large number of trial and errors to', '00:01:22.56 [Speaker1] for a system to learn anything.', "00:01:26.56 [Speaker1] And that's why we don't have self-driving cars.", "00:01:28.40 [Speaker2] That's a big leap for one to the other.", '00:01:30.24 [Speaker2] Okay, so to solve difficult problems, you have to have a lot of', '00:01:42.96 [Speaker2] some way to maybe simulate that problem such that you can do that large scale kind of learning', '00:01:47.68 [Speaker2] that reinforcement learning requires.', '00:01:49.36 [Speaker1] Right, so how is it that most teenagers can learn to drive a car in about 20 hours of practice,', "00:01:57.20 [Speaker1] whereas even with millions of hours of simulated practice, self-driving car can't actually learn", "00:02:4.16 [Speaker1] to drive itself properly? And so obviously we're

Figure 7: Text Transcript

6.2 Extractive Summary

This sub-section showcases the summary generated under the extractive summarization pipeline, under which first the words and sentences are given frequency counts and normalized via TF-IDF, and by providing a certain threshold value, the number of sentences is selected using the text-rank algorithm [the top n sentences are selected], and then they are joined to get one final summary.



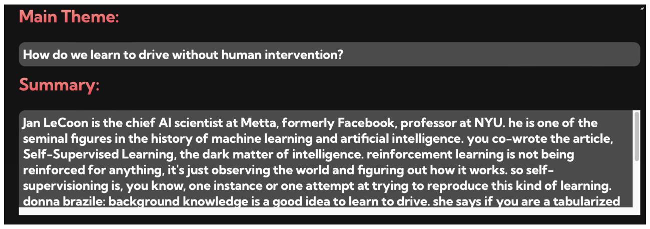
Summary:

Okay, so that to solve difficult problems, you have to have a lot of, uh, human annotations for supervised learning to work, and to solve those difficult problems with the reinforcement learning, you have to have some way to maybe simulate that problem such that you can do that large scale kind of learning that reinforcement learning requires. And there is too many efficient, supervised learning requires many samples for learning anything, and reinforcement learning requires a ridiculous large number of trial and errors to, for, you know, a system to learn anything. And you're saying in order to learn to drive, like the reason humans are able to learn to drive quickly, some faster than others, is because of the background knowledge, they were able to watch cars operate in the world in the many years leading up to it, the physics of basic objects, all that kind of stuff. And you know in advance because of your, you know, understanding of intuitive physics that if you turn the wheel to the right, the car will be out to the right, we're run off the cliff, fall off the cliff, and nothing good will come out of this, right? Right, so how is it that, you know, most teenagers can learn to drive a car in about 20 hours of practice, whereas, uh, even with millions of hours of simulated practice, self-driving car can't actually learn to drive itself properly. So the most popular approaches to machine learning today are, or PyDimes, I should say, are simplevised learning and reinforcement learning. But if you are a sort of, you know, tabularized reinforcement learning system that doesn't have a model of the world, you have to repeat falling off this cliff thousands of times before you figure out it's a bad idea. And it's quite obvious for a lot of people that, you know, the immediate response you get from, many people is, well, you know, humans use their background knowledge to learn faster. His second time on the podcast, he is the chief AI scientist at Metta, formerly Facebook, professor at NYU, touring award winner, one of the seminal figures in the history of machine learning and artificial intelligence, and someone who is brilliant and opinionated in the best kind of way, and so is always fun to talk to. This type of learning is not learning a task, it's not being reinforced for anything, it's just observing the world and figuring out

Figure 8: Text-to-Text Summary

6.3 Abstractive Summary

This sub-section showcases the summary generation as well as routing the theme of the text to better understand the context of the audio file given as input. First, using T5 Tokenizer, the new sentences are created, and then, using the model that we trained by fine-tuning SimpleT5 under the news-summary dataset, the pipeline predicts the theme of the content to generate a more relevant and contextual theme for the input audio file.



Main Theme:

How do we learn to drive without human intervention?

Summary:

Jan LeCoon is the chief AI scientist at Metta, formerly Facebook, professor at NYU. he is one of the seminal figures in the history of machine learning and artificial intelligence. you co-wrote the article, Self-Supervised Learning, the dark matter of intelligence. reinforcement learning is not being reinforced for anything, it's just observing the world and figuring out how it works. so self-supervisioning is, you know, one instance or one attempt at trying to reproduce this kind of learning. donna brazile: background knowledge is a good idea to learn to drive. she says if you are a tabularized

Figure 9: Abstractive Summary

7 RESULT ANALYSIS

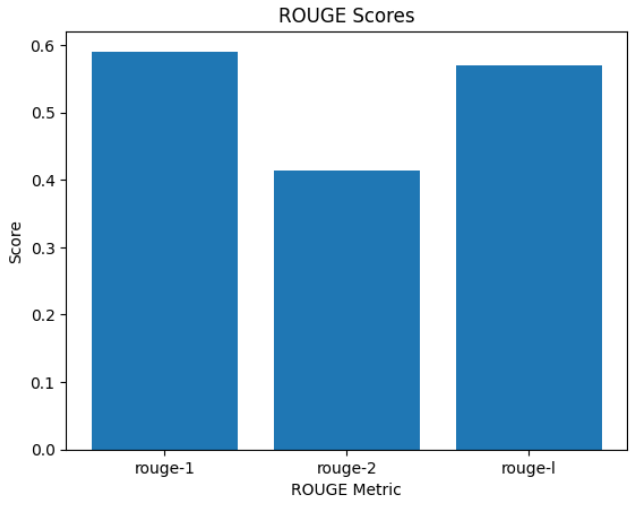


Figure 10: T5 Summarization Model: Precision-Recall Curve

Figure 10 plot is a precision-recall curve for a summarization model trained with T5. This curve helps visualize the trade-off between two key metrics: precision (avoiding irrelevant summaries) and recall (capturing important information). The position of the curve provides insights into the model's ability to generate accurate and informative summaries.

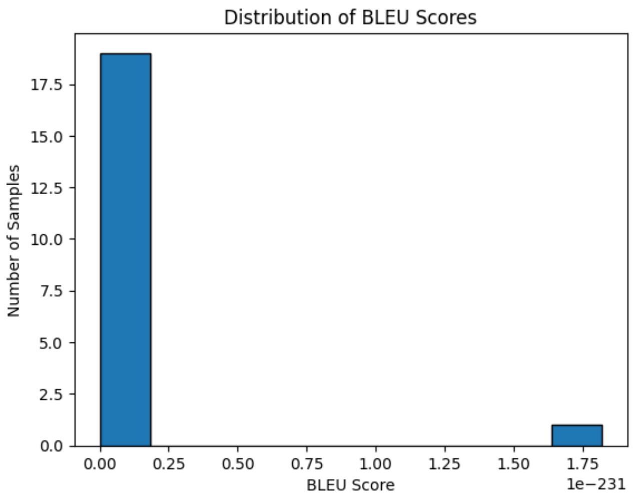


Figure 11: T5 Summarization Model: BLEU Curve

Figure 11: The BLEU score is a common metric for evaluating summaries, but it doesn't directly map to human judgment of quality. This curve helps visualize the trade-off between two factors, precision and recall, potentially influencing the BLEU score.

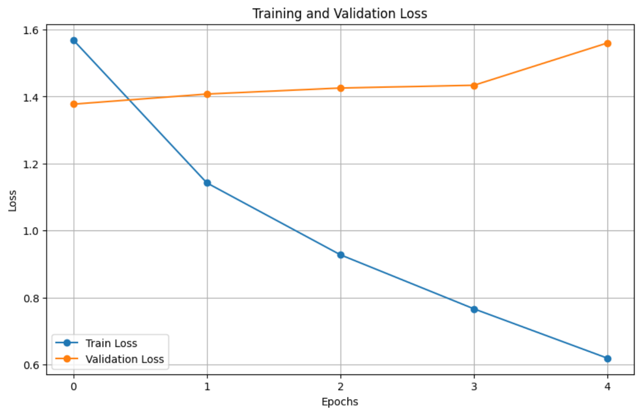


Figure 12: Training and Validation Loss

The figure 12 is a graph that shows the training and validation loss. The graph is a comparison between epochs and their respective loss with respect to training and validation.

8 CONCLUSION AND FUTURE SCOPE

Our work showcases the benefits of natural language processing (NLP) and what can be accomplished using transfer learning. Our approach offers consumers a number of benefits, including increased trust, contextual comprehension, and ability to make well-informed judgments. Given that the research has the potential to produce successful solutions that advance both human and corporate goals, several industrial sectors are keen to investigate this area of study. To create succinct and information-driven summaries, there is room for development in this area, for instance. Real-time processing and customized summary production based on certain meeting kinds are two examples.

ACKNOWLEDGMENTS

We want to thank our project guide, Mr Sai Prasad Kashi for his continuous guidance in figuring out a way to implement the proposed model, and thank Mr Vijay Gopal Jagadam and Mrs Subha our esteemed advisors, for all the guidance and instructions they provided us in the construction of paper and implementation of the model— and our family members for their continuous engagement and support.

REFERENCES

[1] Rajat Verma; Sparsh Gupta; Shubh Sharma; Tanishq Agarwal; Mahesh A.M. Automated meeting minutes generator. *Journal of Emerging Technologies and Innovative Research*, 9(1):2349–5162, January 2022.

[2] K.S Umadevi; R.Jagadesh Kannan; Romansha Chopra; Nivedita Singh; Likitha Aruru. Text summarization of spanish documents. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI) - IEEE*, September 2018.

[3] Anna N; Ond˘rej B. Towards automatic minuting of meetings. *Charles University, Institute of Formal and Applied Linguistics*, August 2019.

[4] Muskaan Singh; Tirthankar Ghosal; Ondrej Bojar. An empirical analysis of text summarization approaches for automatic minuting. *ACL Anthology, Charles University, Czech Republic*, 2021.

[5] Hervé Bredin. pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe. 2023.

[6] Gabriel M; Steve R; Jean C. Extractive summarization of meeting recordings. *Edinburg Research Archive*, June 2012.

[7] Priya Dr. Shashikala P; Vinutha; Arpita; Prema. A novel method for an intelligent based voice meeting system using machine learning. *International Research Journal of Engineering and Technology (IRJET)*, 10(5), May 2023.

[8] Nazim Dugan, Cornelius Glackin, Gérard Chollet, and Nigel Cannings. Intelligent Voice ASR system for Iberspeech 2018 Speech to Text Transcription Challenge. pages 272–276, November 2018.

[9] Jaisal Shah; Neelam Jain. Advances in automatic meeting minute generation. *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, 3:2581–9429, February 2023.

[10] Vishnuprasad;Paul Martin; Salman Nazeer; Prof.Vydehi K. Meeting summarizer using natural language processing. *International Journal for Research in Applied Science and Engineering Technology*, 11, June 2023.

[11] AAKASH SRIVASTAVA; KAMAL CHAUHAN; HIMANSHU DAHARWAL; NIKHIL MUKATI; PRANOTI SHRIKANT KAVIMANDAN5. Text summarizer using nlp (natural language processing). *IRE Journals*, 6:2456–8880, July 2022.

[12] Colin Raffel; Noam Shazeer; Adam Roberts; Katherine Lee; Sharan Narang; Michael Matena; Yanqi Zhou; Wei Li; Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, January 2022.

[13] Swapnil Waghmare; Chaitanya Pathak; Raj Kshirsagar; Suyog Malkar. Business meeting summarization using natural language processing(nlp). *International Journal for Research in Applied Science and Engineering Technology*, 05, July 2021.

[14] Shashi Bhushan TN Md Tahmid Rahman Laskar; Xue-Yong Fu; Cheng Chen. Building real-world meeting summarization systems using large language models: A practical perspective. *Industry Track*, page 343–352, September 2023.

[15] Yashar Mehdad, Giuseppe Carenini, Frank Tompa, and Raymond T. Ng. Abstractive meeting summarization with entailment and fusion. pages 136–146, August 2013.

[16] Hamza Shabbir Moiyadi; Harsh Desai; Dhairya Pawar; Geet Agarwal; Nilesh M.Patil. Nlp based text summarization using semantic analysis. *International Journal of Advanced Engineering, Management and Science (IJAEMS)*, 2, October 2016.

[17] Dikshita Kambri Pallavi Lodhi, Shubhangi Kharche and Sumaiya Khan. Business meeting summarisation system. *2022 2nd Asian Conference on Innovation in Technology (ASIANCON) - IEEE*, May 2022.

[18] Mrs.Sheetal Patil, Avinash Pawar, Siddhi Khanna, Anurag Tiwari, and Somay Trivedi. Text summarizer using nlp (natural language processing). 12:2021, November 2022.

[19] Alexis Plaquet and Hervé Bredin. Powerset multi-class cross entropy loss for neural speaker diarization. 2023.

[20] Srishti Subhash Chandra Prasad;. Business meeting summary generation using natural language processing (nlp). *NORMA eResearch @NCI Library: School of Computing National College of Ireland*, January 2021.

[21] Nitish Singh Rajpurohit; Srujan SP; Tejas Panagar TK; K Padma Priya. Automated generation of minutes of meeting using machine learning. *INTERNATIONAL JOURNAL OF ADVANCE RESEARCH AND INNOVATIVE IDEAS IN EDUCATION - IJARIIE-ISSN(O)-2395-4396*, 2023.

[22] Chetana Varakantham; J. Srinija Reddy; Uday Yelleni; Madhumitha Kotha; Dr P.Venkateswara Rao. Text summarization using nlp. *Journal of Emerging Technologies and Innovative Research*, May 2022.

[23] Mr. Riyazahmed Jamadar; Mehul Pawar; Pavan Karke; Amogh Sonar; Yashshri; Sushant Sharavagi. Automatic speech recognition: Speech to text converter. *International Research Journal of Modernization in Engineering Technology and Science - IRJMETS*, 5, May 2023.

[24] Aryan Jha; Sameer Temkar; Preetam Hegde; Navin Singhaniya. Business meeting summary generation using nlp. *ITM Web of Conferences*, 44, May 2022.

[25] Alec Radford; Jong Wook Kim; Tao Xu; Greg Brockman; Christine McLeavey; Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *OpenAI*, December 2022.

[26] Neslihan Akar; Metin Turan. A general approach for meeting summarization: From speech to extractive summarization. *Review of Computer Engineering Research*, 9(2), 2022.

[27] Virgile Rennard; Guokan Shang; Julie Hunter; Michalis Vazirgiannis. Abstractive meeting summarization:a survey. *arXiv Computation and Language*, December 2023.