Automated Extraction and Augmentation of Key Information from Audio using Speech Recognition and Text Summarization

S Swaroop Kaushik[1]
Bangalore, India

Sanjit Kangovi[2]
Bangalore, India

Abstract:- This Audio lectures and speeches contain a wealth of valuable information, but reviewing and extracting the key points can be tedious and time-consuming. This paper presents an automated system that uses speech recognition and text summarization techniques to identify and summarize the most salient content from spoken presentations. Audio is first transcribed to text via a speech recognition engine. The resulting text is then processed by an extractive summarization algorithm based on term frequency-inverse document frequency (TF-IDF) to extract the most important points. These summarized points can optionally be used to generate relevant supplementary URLs that provide additional context or resources related to the topics covered. This system was developed to enable quick review of lectures and speeches by automatically delivering condensed, relevant summaries.

Keywords:- *Speech Recognition, Extractive Summarization, TF-IDF, URLs.*

I. INTRODUCTION

The COVID-19 pandemic had compelled a widespread transition from traditional in-person classrooms to online education platforms. However, many students struggle to attend or consistently stay focused during virtual classes. Missing sessions or having difficulty concentrating can hinder learning and retention when classes are held remotely. The unique challenges posed by online learning environments require adapted solutions to support students as education moved online amid the pandemic.

Amidst the global shift to online education, the impact has been significant, affecting approximately 1.2 billion children across 186 countries, as highlighted by a report from UNESCO. To address these issues, this paper presents a system that utilizes natural language processing (NLP) techniques to enhance the online education experience.

NLP allows computer algorithms to analyze and comprehend human language. Its applications range from handwriting recognition and speech recognition to creating chatbots and automatic text summarization.

In the context of online education, automatic speech recognition (ASR) plays a crucial role. ASR algorithms possess the ability to convert spoken speech into a written format, allowing for accurate translation of verbal communication into textual form. It is important to note that ASR differs from voice recognition, which focuses on identifying an individual's specific voice.

Meanwhile, text summarization condenses lengthy documents into concise overviews by extracting main points. By generating summarized content, it becomes easier to review online lectures and workshops.

By integrating ASR and text summarization, this paper leverages NLP to mitigate the difficulties of online learning. Automated transcription of classes combined with summarized notes provides students and professionals with streamlined, digestible information. This assists comprehension and retention as education continues adapting to a virtual landscape.

The paper exemplifies how NLP and AI can enhance remote collaboration and learning during an unprecedented shift to online platforms. With customized tools to target unique challenges, technology can facilitate engagement, understanding, and memory despite the limitations of distance learning.

There are two main methods in order to summarize a given text, that is

➢ *Abstractive Summarization*
In abstractive summarization, the given source text document is paraphrased and shortened as required. With an abstraction algorithm, grammatical inconsistencies can be avoided as compared to extractive summarization methods. The abstractive summarization uses trained data to create new phrases and sentences that provide the most important information from the text.

➢ *Extractive Summarization*
Extractive summarization algorithms include extracting the key phrases from the source text document and then integrating them to generate a summary. There are no changes made to the words or phrases in the source text and the summary is generated according to the given metrics. Extractive summarization algorithms do not require an exhaustive set of training data and are comparatively less complex than abstractive summarization techniques and thus are widely popular.

This paper introduces a method wherein real-time audio is converted into textual data by a speech-to-text module. Further, the textual data is used as input to the text summarizer. The text summarizer uses TF-IDF to generate a shortened or summarized form of the textual data using extractive summarization. Furthermore, based on the keywords from the summary, a web scraping algorithm is used to search and collect data off the web related to the keywords and fetch the respective URLs. This method would be considerably advantageous to minimize human effort in time-crunch situations like speeches or lectures where summarizing the information on the go might pose a challenge.

This paper is organized into 5 sections. Section II provides a literature review, examining existing research related to the paper's domain. This is followed by Section III, which outlines the theoretical underpinnings of the algorithms used. The system procedure is then explained in Section IV. Section V describes the simulation methodology, while Section VI discusses the results obtained. Finally, the paper concludes by considering potential applications and directions for future work.

II. LITERATURE SURVEY

The objective of this literature survey is to evaluate the methodologies present to implement text summarization and to note the ways in which problems such as huge dimensionality, poor transparency, and unavailability of proper datasets have been solved.

One notable approach is the use of Generative Adversarial Networks (GANs) [1]. This framework consists of a generator and a discriminator. The generator employs reinforcement learning techniques to generate rewarded summaries, overcoming exposure bias and non-differentiable metrics. The discriminator acts as a text classifier, distinguishing between machine-generated and human-generated summaries. Through a minimax game optimization process, the proposed GAN-based approach achieves high-quality abstractive summaries. The generator incorporates bidirectional LSTM encoders, attention-based LSTM decoders, and a switching pointer-generator network. The discriminator utilizes a CNN architecture with pooling and classification layers. This research significantly advances abstractive summarization methods, improving the quality and effectiveness of summary generation.

In a research paper [2], an abstractive text summarization approach using sequence-to-sequence Recurrent Neural Networks (RNNs) is presented. Originally designed for machine translation tasks, this model is adapted for summarizing food reviews. It employs a two-layered RNN network with LSTM cells and attention mechanisms applied to the target text. By maximizing the conditional probability of the target text sequence, the model generates comprehensive summaries. However, this approach requires a substantial amount of training data and entails long computational time and resource-intensive hardware. Nevertheless, it demonstrates the effectiveness of sequence-to-sequence RNNs in abstractive text summarization, specifically in the domain of food reviews.

In addition to these approaches, other methods have been proposed. In [3], a supervised learning approach using K-nearest neighbors (KNN) for extractive text summarization is presented. The authors modify the traditional KNN model to consider feature and feature value similarity, aiming to enhance the summarization process. By mapping texts to numerical vectors and assigning different weights to selected neighbors, the model generates summaries based on the semantic relations between words. However, this approach requires pre-trained tables for each domain during training, adding manual effort to the process. Nonetheless, it offers a promising technique for extractive summarization with improved similarity measures and a modified KNN algorithm.

Furthermore, [4] proposes a Pointer-Generator Abstractive Model with Part of Speech Features, combining word vectors, parts of speech, CNN, and bi-directional LSTM. TF-IDF is explored for keyword relevance in summarization [5], and K-means clustering with TF-IDF is introduced for sentence clustering and summarization [6]. Each method has its own strengths and weaknesses, with abstractive summaries being more human-like but requiring extensive training data, while extractive methods are more adaptable but often lack natural phrasing. To enhance extractive summarization, additional attention layers and network models have been suggested.

Considering all the surveyed methods, it can be inferred that the extractive summarization along with the TF-IDF algorithm would be suitable for the chosen problem statement to implement text summarization. The chosen algorithm will not require large datasets to train the model and thus can make the model applicable to all the subject classes. It will also have the advantage of time complexity in comparison to the abstractive summarization methodology.

III. THEORY

A. NLTK

NLTK is a standard Python library that comes with prebuilt functions and utilities for convenience of usage and implementation. It is among the most used NLP and computational linguistics libraries. For our experiments, we used the nltk library for data pre-processing such as Tokenization. Tokenization in NLP is the process of breaking down or separating sentences into smaller units called tokens. We used two types of tokenization for our experiments:

Sentence tokenization - Sentence tokenization, performed using the *sent_tokenize* method, involves splitting text into individual sentences. This tokenizer ensures that individual words within a sentence remain intact during the tokenization process.

Word tokenization - The process of segmenting a sentence into individual words is called word tokenization. The *word_tokenize* function, whose result is a list of words, has been used. In machine learning applications, the word tokenization output can be transformed into a data frame to improve text understanding. It can also be used as an input for text cleaning processes that come after, like stemming, punctuation removal, and numeric character removal.

➢ *Additionally, we Employed the following NLTK Functionalities for text Normalization:*

- *Stop Words Removal:*

Stop words are words that occur commonly in a corpus such as “the”, “a”, “an”, “in” etc. that, in order to achieve text normalization, must be eliminated from the corpus.

- *Stemming:*

Stemming is the process of producing morphological variants of a root/base word. It is the reduction of inflection from words. Words with the same origin will get reduced to a form that may or may not be a word. *PorterStemmer* function has been used for stemming.

B. Beautiful Soup and Cloud Client Libraries:

Beautiful Soup is a Python package used for parsing HTML and XML documents. The Cloud Client Libraries in Python are used to access Google Cloud APIs programmatically. The libraries make things easier to grasp by offering high-level abstractions for the API. These libraries have been used for speech processing and web scraping in our experiments to retrieve links to important topics from the summarized text.

IV. PROCEDURE

The proposed methodology consists of a three-step process, which includes the conversion of speech data to text data using the Speech Recognizer class using Python. The second step consists of summarizing the text data to provide a readable and concise summary using NLTK libraries by implementing the Term Frequency- Inverse Document Frequency (TF-IDF) methodology. This is implemented by using extractive methods for summarization. After obtaining the summary, the last step to improve learning is by analyzing the output to provide links to websites relevant to the topic using web scraping.

A. Conversion of raw Data from Speech to Text

The first step involves converting raw speech data into text data using Python's speech recognizer class. The Speech Recognizer library is installed and imported into the class. The Recognizer class utilizes a microphone to recognize and record speech input. The recorded speech data is passed to the Recognizer class's *recognize_google* method, which converts the audio file into text.

B. Generation of a Summary using TF-IDF Methodology

The second step focuses on generating a readable and concise summary using the TF-IDF methodology and the NLTK libraries. The proposed method for text summarization encompasses the following steps:

➢ *Data Pre Processing*

The data preprocessing involves the following methods:

- *Tokenization –*

In tokenization the text data is divided into smaller parts called tokens. Two types of tokenization are employed: word tokenization, which splits sentences into words using the *word_tokenize* method, and sentence tokenization, which splits text into individual sentences using the *sent_tokenize* method. Tokenization aids in pattern finding, stemming, and lemmatization processes.

- *Stemming –*

Stemming is performed using the *PorterStemmer* algorithm to produce morphological variants of root/base words. This process reduces redundancy, as the word stem and their inflected/derived words often convey the same meaning.

➢ *Creating a Frequency Table*

Finding the most or least common terms based on the algorithm's requirements and calculating word frequencies are crucial. As a result, a dictionary including all of the terms in our corpus as keys and their frequency of occurrence as values was created.

➢ *Calculation of Term Frequency (TF)*

This stage involves calculating each word's term frequency (TF) and creating a matrix. The term "TF" denotes the frequency with which a word occurs in the phrase relative to its entire word count. Words with comparable frequencies have similar TF scores, as can be seen by comparing the frequency table to the TF matrix.

➢ *Creating a Table for Documents per Word*

This again a simple table that helps in calculating the IDF matrix. The number of sentences that contain a particular word is calculated, which is then used to generate the documents per word table.

➢ *Calculation of Inverse Document Frequency (IDF)*

In this step, the IDF score is determined by taking a log10 of the number obtained by dividing the total number of sentences by the number of sentences that contain the word. The IDF matrix quantifies the uniqueness of words in a paragraph.

➢ *Calculation of TF-IDF*

The TF-IDF score is obtained by multiplying the TF and IDF values for each key-value pair from the TF and IDF matrices. The resulting TF-IDF matrix assigns weightage to words within a document.

➢ *Scoring the Sentences*

We used the TF-IDF score to assign a paragraph's weight based on words in a sentence. This was performed by taking the individual sentences from tokenized sentences and computing the sentence score to determine which sentences are the most essential. Following the computation of scores, the user-provided retention rate was used to determine which sentences rank highest and are summarized accordingly.

➢ *Determining the Threshold*

The threshold, which determines the inclusion or exclusion of sentences in the summary, can be found in various ways. In this paper, the average sentence score is calculated to generate the threshold.

➢ *Generation of Summary*

The final summary is generated by considering the sentences whose scores exceed a certain threshold, ensuring an accurate and concise summary.

C. Generation of Relevant URLs

The third step involves generating relevant URLs related to the summarized text. Web scraping techniques are employed to extract links that are relevant to the generated summary. This process entails two parts: the crawler and the scraper. The crawler searches the web by following links, while the scraper is designed to extract data from websites. The Beautiful Soup and Google libraries are utilized for this purpose. The number of links to be generated can be controlled by adjusting the values of 'num,' 'stop,' and 'pause.' values.

V. SIMULATION

To evaluate the performance of the summarization model, several simulations were conducted using Python and Google Colab. Speech recordings on three distinct topics - history, science, and geography - were provided, mimicking classroom lectures of 903, 1478, and 760 words respectively. For each topic, the threshold coefficient of the model was manipulated across three values ranging from 0.3 to 1.1. With each threshold value, the model generated a summary and the output word count was calculated. It was observed that lower threshold coefficients resulted in summaries with higher word counts, while increasing the threshold reduced the word count and summary length. This is because the model passes only those sentences that have a sentence score value exceeding the specified threshold. Using higher threshold values ensures that only the most salient sentences are included, generating more precise and concise summaries. The results demonstrate that the summarization model can effectively adjust summary brevity by tuning the threshold parameter. Across the diverse lecture topics, the model successfully produced condensed summaries while preserving the core content.

Additionally, the web scraping model was implemented to fetch relevant links related to the summarization topics. The number of generated links could also be varied as needed.

This simulation study provides promising indications that the approach can condense long-form speech into concise summaries of key points through automated extraction. Although Google Colab was utilized, the model can be implemented in any Python-based integrated development environment, such as Jupyter Notebook.

Table 1 Simulation Result

Type of Input	No. of Input Words	Avg. Sentence Score	Threshold Coeff.	No. of Output Words
History	903	0.08830773536	0.7	312
		0.08830773536	0.3	432
Science	1478	0.07899019456	0.9	699
		0.07899019456	1.2	220
Geography	760	0.07737628511	0.8	483
		0.07737628511	1.1	160

VI. RESULTS

In multiple trials that were conducted with different test cases, output was obtained by conversion of raw data from speech to text, generation of a summary using TF-IDF methodology, and generation of relevant links related to the topic.

The model was verified to convert the raw data which is either in the form of live speech or recorded audio data to text, post which it could make a summary of the converted speech using the TF-IDF methodology where the raw data of about 800- 1500 words was reduced down to a summary 200-500 words by varying the threshold coefficient, and finally, with crawler and scraper, relevant URLs were generated. This would simplify the understanding process for the user and help with in-depth analysis.

VII. CONCLUSION

In conclusion, this paper demonstrates the potential for using speech recognition and extractive text summarization techniques to automatically extract key information from audio lectures and speeches. The speech-to-text conversion allows the textual content to be processed by the TF-IDF text summarizer, identifying the most salient points. These summaries can then be used to recommend relevant supplementary materials through URL generation, enhancing the knowledge gained from the original audio source. This approach could be useful for improving accessibility of audio materials and enabling quick review of lectures or speeches. Further work on improving speech recognition accuracy and summarization quality could make this solution even more effective. Overall, this paper shows promising techniques for automatically extracting and augmenting important information from real life audio sources. Looking ahead, this system could potentially be integrated into a dedicated hardware device to enable portable, automated lecture or speech summarization. Users could record audio through the device and easily review the generated summary and recommendations immediately after events. Developing this into a handheld gadget would further increase the accessibility and convenience of the solution.

REFERENCES

[1]. Liu, Linqing, et al. "Generative adversarial network for abstractive text summarization." Proceedings of the AAAI conference on artificial intelligence. Vol. 32. No. 1. 2018.

[2]. A. K. Mohammad Masum, S. Abujar, M. A. Islam Talukder, A. K. M. S. Azad Rabby and S. A. Hossain, "Abstractive method of text summarization with sequence to sequence RNNs," 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kanpur, India, 2019, pp. 1-5, doi: 10.1109/ICCCNT45670. 2019.8944620.

[3]. T. Jo, "K nearest neighbor for text summarization using feature similarity," 2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE), Khartoum, Sudan, 2017, pp. 1-5, doi: 10.1109/ICCCCEE.2017.7866705.

[4]. Ren, Shuxia, and Zheming Zhang. "Pointer-Generator Abstractive Text Summarization Model with Part of Speech Features." 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS). IEEE, 2019.

[5]. Qaiser, Shahzad, and Ramsha Ali. "Text mining: use of TF-IDF to examine the relevance of words to documents." International Journal of Computer Applications 181.1 (2018): 25-29.

[6]. Khan, Rahim, Yurong Qian, and Sajid Naeem. "Extractive based text summarization using k-means and tf-idf." International Journal of Information Engineering and Electronic Business 10.3 (2019): 33.

[7]. J. N. Madhuri and R. Ganesh Kumar, "Extractive Text Summarization Using Sentence Ranking," 2019 International Conference on Data Science and Communication (IconDSC), 2019, pp. 1-3, doi: 10.1109/IconDSC.2019.8817040.

[8]. P. M. ee, S. Santra, S. Bhowmick, A. Paul, P. Chatterjee and A. Deyasi, "Development of GUI for Text-to-Speech Recognition using Natural Language Processing," 2018 2nd International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech), 2018, pp. 1-4, doi: 10.1109/IEMENTECH.2018.8465238.

[9]. J. Zenkert, A. Klahold and M. Fathi, "Towards Extractive Text Summarization Using Multidimensional Knowledge Representation," 2018 IEEE International Conference on Electro/Information Technology (EIT), Rochester, MI, USA, 2018, pp. 0826-0831, doi: 10.1109/EIT.2018.8500186.

[10]. D. Inouye and J. K. Kalita, "Comparing Twitter Summarization Algorithms for Multiple Post Summaries," 2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing, Boston, MA, USA, 2011, pp. 298-306, doi: 10.1109/PASSAT/SocialCom.2011.31.

[11]. G. Xu, Y. Meng, X. Qiu, Z. Yu and X. Wu, "Sentiment Analysis of Comment Texts Based on BiLSTM," in IEEE Access, vol. 7, pp. 51522-51532, 2019, doi: 10.1109/ACCESS.2019.2909919.

[12]. H. P. Luhn, "The Automatic Creation of Literature Abstracts," in IBM Journal of Research and Development, vol. 2, no. 2, pp. 159-165, Apr. 1958, doi: 10.1147/rd.22.0159.

[13]. Gupta, V., & Lehal, G.S., (2010). A survey of text summarization extractive techniques. Journal of emerging technologies in web intelligence, 2(3), pp.258-268.

[14]. Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B. and Kochut, K., (2017). A brief survey of text mining: Classification, clustering and extraction techniques. arXiv preprint arXiv:1707.02919.