2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)
Bangalore, India. Oct 7-9, 2022

Audio Synthesis Translation and Auto-Summarization (ASTA)

Jivin Varghese
Department of Computer Engineering
Don Bosco Institute of Technology
Mumbai, India
jivinvarghese007@gmail.com

Pakshal Ranawat
Department of Computer Engineering
Don Bosco Institute of Technology
Mumbai, India
ranawatpakshal31@gmail.com

Ruvin Rodrigues
Department of Computer Engineering
Don Bosco Institute of Technology
Mumbai, India
ruvin27@gmail.com

Dr. Phiroj Shaikh
Department of Computer Engineering
Don Bosco Institute of Technology
Mumbai, India
phiroj@dbit.in

2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT) | 978-1-6654-6855-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/GCAT55367.2022.9971977

***Abstract*— Availability of time has been a major issue in recent years for mankind. There has always been a huge demand for automation, since it can tremendously decrease time for doing menial tasks. This proposed project focuses on automation of text translation, summarization and speech synthesis which could reduce time required for reading books. In this paper, we present multiple machine learning models that synthesize text into speech and also into summarized text of Devanagari script. The main objective of the project is to conduct proper examination of the existing architecture of the text translation and summarization methodologies and to provide a robust system which is a cumulation of converting PDF files to audio files and also summarization of the PDF and translating into Devanagari text of the summarized English narrative. The architecture is called Audio Synthesis Translation and Auto-summarization (ASTA) and uses multiple models such as RNN sequence to sequence, NMT, Tacotron 2 and Waveglow. In addition to this, we use Google Vision OCR for text extraction from PDF. This system is an integration of multiple machine learning models and works as a pipelined system.**

Keywords*— RNN, LSTM, Keras, TensorFlow, Neural Networks, Tacotron2, NMT, Vision OCR, WaveGlow.

I. Introduction

Neural Networks is a network of multiple networks which consists of interconnected nodes. It can be used to identify and classify patterns just like neurons in a human brain by ensuring the correctness of fit for any given observation. Using Recurrent neural networks ensures a better approach to traditional machine learning and natural language processing methodologies.

To have an infrastructure or a pipeline model to cater the need of creating audio files of the textual content present in this digital era where portability, accessible content and ease of use is always a priority. Whereas the need of summarization and translation to create a concise coherent text for understanding the essence of the files can help in creating a better under- standing of the content. There are multiple techniques present for solving this scenario and we decided to approach the problem with Recurrent Neural Networks. An encoder-decoder sequence to sequence model is a type of recurrent neural network architecture that is used to predict outputs such as text-translation and summarization in natural language processing. The demand for such architectures thus, has been ever increasing. Moreover, the technology to make the availability of custom audio files according to a person's requirement is generally less available according to their linguistic backgrounds. Generating Audio files out of a PDF has high demand. It is a boon especially for the people who have ailments like the visually impaired. Additionally, a PDF file that can be summarized and translated into a different language can assist people from various linguistic backgrounds understand ideologies that they previously couldn't.

The models for summarizing text which are widely used are extractive in nature which extract sentences from the input to create a summary. In the proposed architecture, the problem statement is approached in an abstract nature such as to create an entire new sentence from the input data which is coherent. Due to the research in this domain being limited, it still remains a challenging task despite extensive research. The proposed project is split into two modules where a PDF is given as input and the output can either be an audio file generated from the text in the PDF or a concise summary of the PDF in the Devanagari script.

II. Literature Survey

The system has been divided into 2 parts and further subdivided into 4 modules in total and the literature review is done accordingly.

A. Text Extraction

In the research paper [1] written by Rifiana Arief et al and [2] written by D Vaithiyanathan, Manigandan Muniraj, it was proposed that Google Vision OCR that is used for recognition of characters was far more optimal than other OCR engines. It handles the text extraction robustly with far more effective algorithms.

B. Text-To-Speech Synthesis

The traditional text to speech models do not overcome the concern of generating natural sounding speech. In the paper [6] written by Jonathan Shen et al have proposed tacotron 2 for speech synthesis. Tacotron 2 is a sequence to sequence model for converting character sequence to

amplitude spectrograms. It is an end to end model where it maps the sequences of source text to mel-scale spectrograms using a feature prediction network. Furthermore, it is followed by a modified WaveNet model which takes in the mel-scale spectrogram and synthesizes time-domain waveforms by acting as a vocoder. They have proposed a fully neural TTS system. It is a single neural network model which has overcome the traditional models of speech synthesis processes. Tacotron 2 is more effective at generating natural sounding speech.

C. Text Extraction

Text Summarization is broadly classified into two types:

1) *Abstractive Summarization*
2) *Extractive Summarization*

Abstractive summarization not only extracts important phrases from source text but also generates new sentences when provided with source text to the model to create a concise summary whereas extractive summarization only extracts sentences from the given source text and uses the same for generating the summary.

In the paper [4] written by Ramesh Nallapati et al they discuss how to summarize the text using the RNN approach. Abstractive summarization is carried out using the deep learning technique called Long Short Term Memory (LSTM) which is a type of Recurrent Neural Network Algorithms (RNN). RNN is an encoder-decoder sequence to sequence model which consists of multiple hidden states with their corresponding weights.

D. Text-To-Text Translation

The paper [5] written by Mike Schuster et al talk about how to input text to corresponding output text and the strength of NMTs. English text is converted to Devanagari Text using GNMT (Google Neural Machine Translation). GNMT is a modified version of a neural machine translation system. Other Algorithms as compared to GNMT have many flaws such as slow training speed, problems while dealing with unseen words whereas GNMT solves this problem using an 8 layer LSTM RNNs. The bottom layer of decoder of the LSTM has an attention layer which is connected to the top layer of encoder for achieving parallelism.

III. Proposed Architecture

There are 4 main modules in the proposed architecture which include Text Extraction, Text-to-Speech Synthesis, Summarization and Text-to-Text Translation.

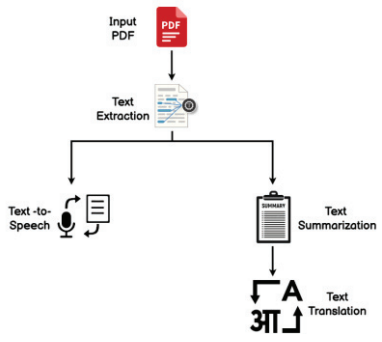

Fig. 1. High-Level System Architecture

A. Text Extraction

Presently, there are multiple optical recognition character models. Google Vision OCR is one of the features of Google cloud vision API which are used in Optical Character Recognition. Vision OCR is implemented for text extraction from the selected PDF. It provides the best quality when it comes to obtaining clear and precise text data in a number of languages. In this module, Vision OCR acts as a middle-ware to detect and extract text from the pages of the PDF files which were converted to images.

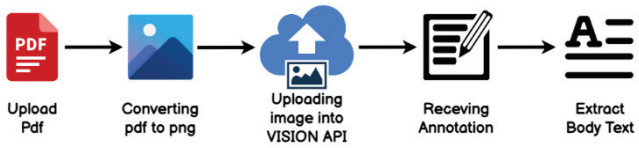

Fig. 2. Text Extraction Process

B. Text-To-Speech Synthesis

For this module, we used Tacotron 2 and WaveGlow for audio synthesis. Tacotron 2 model is based on sequence to sequence Recurrent Neural Network with attention mechanism. Tacotron 2 makes use of basic construct- ing pieces, along with simple LSTM and convolutional layers withinside the encoder and decoder. A sampling rate of more than 16 kHz is needed, since below 16 kHz the speech generated is poor in quality and contains distortions. WaveGlow is a model that samples from a distribution to generate sounds. A zero mean spherical Gaussian is used to take samples with the similar shape as our output. These samples are then passed through a number of layers that transfigures the simple distribution into the expected distribution and the output is generated using concatenative synthesis in the audio form.

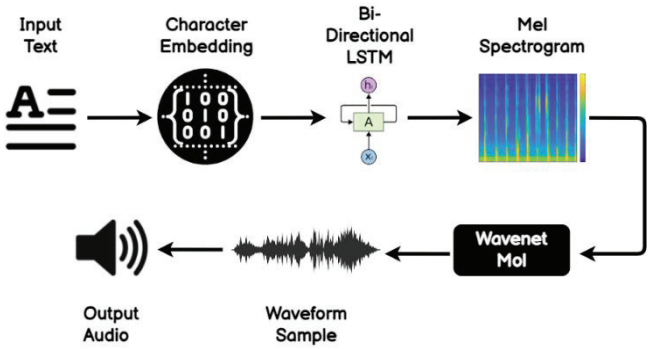

Fig. 3. Tacotron2 Architecture

C. Text Summarization

For the text Summarization module, the model proposed below is an 12-layer abstractive summarization model with 2 input layers, 2 embedding layers, 4 encoding LSTM layers, a decoding LSTM layer, an attention layer, a concatenation layer and a dense layer. The input layer 1 is fed with the source text which is to be summarized and passed to the embedding layer to convert into a fixed length vector format. We used stacked LSTM encoders because of the longer length of the input text so as to retain more information of the previous time step. The stacked encoding LSTM layers are fed with a fixed length vector from the Embedding layer 1. The purpose of the additional layers is to synthesize the internal representation from previous layers and construct new representations at high abstraction levels.

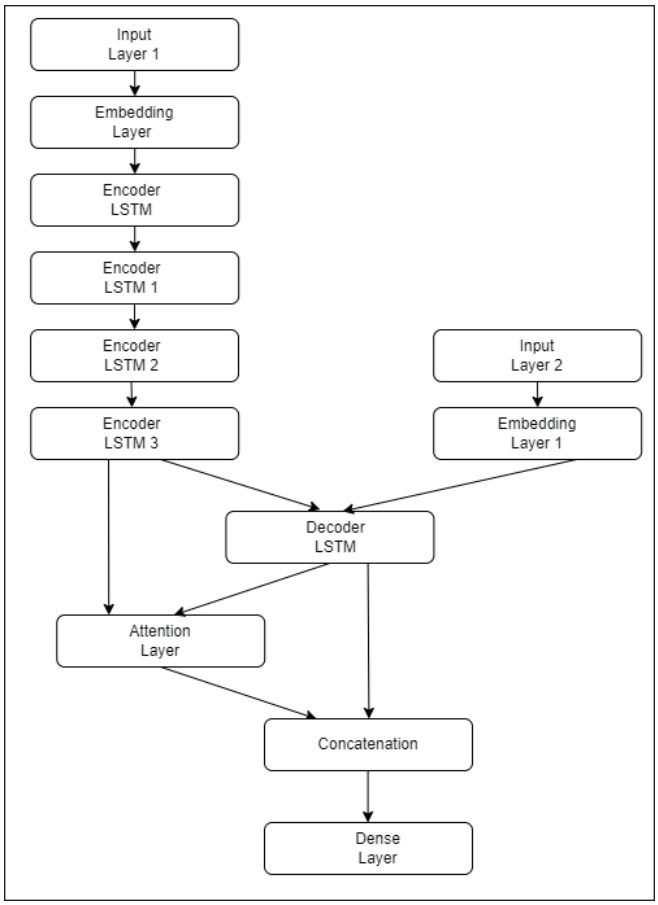

Fig. 4. Text Summarization Proposed Model Architecture

Teacher forcing is used to train the decoder LSTM with expected outputs from the training dataset. The output from the encoder and decoder LSTMs are fed to the attention layer where the attention layer looks for a group of areas in the encoded input sequence where the most crucial data is concentrated. The concatenation layer concatenates the outputs from the attention layer and the decoder LSTM at each time step and passes it to the dense layer. The dense layer processes the output from the concatenation layer in a loop and generates a token until it encounters the END token in the input sequence.

D. Text-To-Text Translation

The model given below is the proposed model that follows the sequence to sequence RNN architecture. The model takes English text as input and Devanagari translated text output is produced. The model is made of 7 seven layers consisting of 2 input layers, 2 embedding layers, 2 lstm layers and a dense layer. Firstly, data preprocessing techniques are applied such as cleaning and tokenization. The cleaned text is passed into the input layer where it accepts the data and passes it to the rest of the network.

The embedding layer receives input sequences with one word for each time step. Each word is encoded into a vector format. The size of the vector is determined by the vocabulary's complexity. We can capture more precise syntactic and semantic word associations using embeddings. The embedding layer projects each word into n-dimensional space. Words with similar meanings are clustered together in this space. The closer two words are, the more related they are. The 2D vector output from the embedding layer is

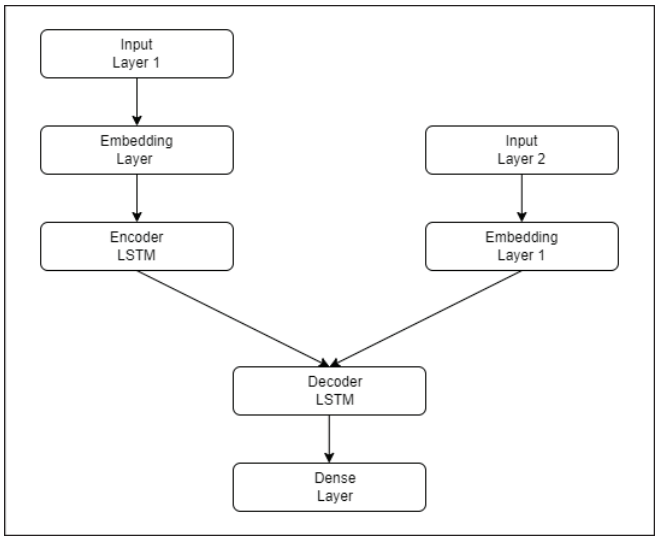

Fig. 5. Text-to-Text Translation Proposed Model Architecture

passed to the encoder LSTM. The input sequence is fed into the encoding layer, which converts it to a fixed-length internal representation. The context from preceding time steps' word vectors is applied to the current word vector in this phase. The initial states of the decoder LSTM are set to the encoder LSTM's final states. The decoder subsequently converts the vector representation from the encoder LSTM to a variable-length target sequence. Teacher forcing strategy is used for the decoder LSTM that uses ground truth as input, instead of the state output from a preceding time step as an input. Finally, a dense layer is used for changing the dimensions of the vector using matrix-vector multiplication.

IV. IMPLEMENTATION

A. Text Extraction

There are two APIs which can help in character recognition namely Text_Annotation and Document_ Text_Annotation. Text_Annotation takes an image file as input and uses multiple image correction algorithms for obtaining better lighting conditions. This API returns a response in json format which consists of the text inside the pages along with the bounding boxes of the respective content. Document_Text_Annotation is used to extract the text from files of not more than 2000 pages.

In the proposed project, Text_Annotation API from Google Vision OCR is used. A PDF file is taken as input after which each page of the PDF is converted to an image as Vision OCR accepts image format only. After sending the images through the API, it returns a response object of the corresponding images which is in json format. The text from these images are then parsed through from the json object.

B. Text-To-Speech Synthesis

Using Pytorch, we used the Tacotron 2 and Waveglow model ensemble to synthesize natural sounding speech without giving additional prosody information. The input of raw transcripts is fed to the Tacotron 2. The input character of every lexeme is converted to mel-spectrogram frames. The mel-spectrogram frames are then passed to WaveGlow which is a flow based model that consumes mel-spectrogram to generate time-domain waveforms. The waveforms were generated by WaveGlow at 22 kHz.

The output generated from this TTS system are audio frames of 11 seconds in duration. Implementation of concatenative speech synthesis, a traditional approach where short audio frames generated are subsequently administered to form one single audio file.

```
[15] from scipy.io.wavfile import write
     from IPython.display import Audio
     def audiofunct(text):
       sequences, lengths = utils.prepare_input_sequence([text])
       with torch.no_grad():
           mel, _, _ = tacotron2.infer(sequences, lengths)
           audio = waveglow.infer(mel)
       audio_numpy = audio[0].data.cpu().numpy()
       return audio_numpy
```

```
from pydub import AudioSegment
rate = 22050
audio = []
n = len(new_list)
for i in range(n):
  audio.append(audiofunct(new_list[i]))
```

Fig. 6. Audio Generation Pt. 1

```
for i in range(n):
  if i == 0:
    write("/drive/My Drive/Colab Notebooks/audio1.wav",rate,audio[0])
  else:
    write("/drive/My Drive/Colab Notebooks/audio2.wav",rate,audio[i])
    sound1 = AudioSegment.from_wav("/drive/My Drive/Colab Notebooks/audio1.wav")
    sound2 = AudioSegment.from_wav("/drive/My Drive/Colab Notebooks/audio2.wav")
    combined_sounds = sound1 + sound2
    combined_sounds.export("/drive/My Drive/Colab Notebooks/audio1.wav", format="wav")
```

Fig. 7. Audio Generation Pt. 2

C. *Text Summarization*

Text summarization has been implemented using 12 layers in total from python's keras library, as well as other packages needed for data preprocessing and design. Abstractive summarization approach is used where the model generates a brief and succinct summary of the main points of the source material. Moreover, the attention mechanism is employed, which focuses on relevant information from the encoded hidden states and aids the decoder in finding the next word in the summary by instructing it where to look in the source words. The two strategies, Teacher forcing and Stacked LSTM encoders increased the amount of abstraction in the resulting summary where, in the teacher forcing strategy, the decoder LSTM is feeded ground truth inputs and the stacked LSTM encoders increases the level of abstraction in the encoded hidden states. The model was trained for 15 hours on the news_summary dataset using more than 120000 sets of source text and summaries. The output from the summarization model is saved onto a text file as well as passed to the text translation model as input.

```
PS C:\Users\Ruvin\Desktop\BE\Module 2> python .\Main.py
The Administration of Union Territory Daman and Diu has revoked its order that made it
compulsory for women to tie rakhis to their male colleagues on the occasion of Rakshabandhan
on August 7. The administration was forced to withdraw the decision within 24 hours of issuing
the circular after it received flak from employees and was slammed on social media.

Files Successfully saved on the Machine
```

Fig. 8. Text extracted given input to summarization module

summary text.txt

```
1  the order made it compulsory for women to tie
2   rakhis to male colleagues. the administration
```

Fig. 9. Summarized Text

D. *Text-To-Text Translation*

With the aid of python's keras library, the text translation module is implemented. The Hindi English_Truncated _Corpus_dataset was used to train the 7-layer model for 16 hours. The model takes the output from the summarization module as input where it passes through the input layer and the embedding layer for data preprocessing and embedding. The embedded words are then sent to the encoder LSTM to be translated. The current embedded vector and the previously calculated hidden state from the previous timestep is multiplied by some weight matrix individually. After adding the out- puts of these two multiplications, a non-linear activation function is used. This is now our next hidden state (h). This process is then repeated until the encoder comes across the END token. Teacher forcing technique is used to train the decoder LSTM where the preceding word from the expected output is sent into the next network instead of the prior prediction generated by the decoder, which considerably aids the training process.

hindi text.txt

```
1  इस व्यवस्था ने स्त्रियों के लिए अपने आदमियों को अपने
2  साथ काम करनेवाले आदमियों को बाँध देना ज़रूरी बना दिया ।
```

Fig. 10. English-to-Hindi Converted Text

V. Result

The initial stage of implementation of the system was getting a hold of the corpus. Two databases were procured for the system one pertaining to summarization and the other to translation. Proceeding ahead text was extracted with the assistance of Google Vision OCR as a base. This text needed to be processed as seen in Figure 2 and for that intention first the text to speech module was implemented.

In the text to speech synthesis process, Tacotron 2 was wielded in order to convert the extracted English text into English audio. Here, at a time 256 characters were converted into 11 second audio frames, which were concatenated to form the final audio file. The module showed that one can hear slight loss in the prosody of the audio speech, this was due to the application of concatenative synthesis.

The next course of action implemented was working on the summarization module, wherein encoders such as stacked LSTM of 4 LSTM encoders were taken into consideration. Abstractive text summarization models are encountered with problems for producing summary which has issues pertaining to relationship between entities in the input dataset which causes problems while connecting the two or more entities On account of bridging that gap, stacked LSTM was used, seeing as it supported the retention of the abstract of longer sentences and also generated more cogent and logical sequences.

Since there is no proper evaluation metrics for Abstractive Summarization, we used ROUGE score metrics which is used for extractive summarization. The Evaluation metrics for summarization model that we used is ROUGE-1 and ROUGE-L. The overlap of each word between a system and the referred summary is known as ROUGE-1. ROUGE-L stands for data based on the Longest Common Subsequence. LCS discovers the longest occurring sequence by taking into consideration the similarities in the sentence.

TABLE I. SUMMARY EVALUATION SCORE

Evaluation Metric	*Score*
ROUGE-1	20.74
ROUGE-L	20.91

Whilst coming to the findings of this model, we were able to generate a single line summary but unfortunately couldn't capture the entire essence of the passage. Also, owing to the corpus not being apt due to it containing single line summaries, the end product was a single line summary regardless of the size of the text that was feeded as the input. The output for English to Hindi translation for the model was semantically correct. Due to the dataset being small, some words which are uncommon are misinterpreted. This can be further improved by using larger dataset and better hardware for longer training durations.

VI. CONCLUSION

In this paper, we describe in detail the implementation of a pipeline model for converting text from PDF files to audio files and also implementing a translation for the summarized text from the PDF. The execution of the summarization module displayed that the resulting output obtained regardless of the size of the text input was a single line output. This can be further worked upon to create a multiline summary for greater size of inputs with the help of better and larger datasets. The Text-to-Speech module results in natural sounding speech however due to concatenative synthesis, we can hear slight loss in the prosody of the audio speech. Lastly, the Translation model was trained for converting the English text into Devanagari text. The result pertaining to the Devanagari text was found to be semantically accurate.

REFERENCES

[1] R. Arief, A. B. Mutiara, A. B. Mutiara, and Hustinawaty, "Automated extraction of large scale scanned document im- ages using google vision ocr in apache hadoop environment," (IJACSA) International Journal of Advanced Computer Science and Applications, January 2018.

[2] M. M. D Vaithiyanathan, "Cloud based text extraction using google cloud vision for visually impaired applications," 11th International Conference on Advanced Computing (ICoAC), 2019.

[3] K. Divya, K. Sneha, B. Sowmya, and G. S. Rao, "Text summarization using deep learning," International Research Journal of Engineering and Technology (IRJET), May 2020.

[4] R. Nallapati, B. Zhou, C. dos Santos, Ça˘glar G˙Ulçehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence rnns and beyond," Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, February 2016.

[5] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, ukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," Arxiv, Oct 2016.

[6] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y.Zhang, Y.Wang, R.Skerry-Ryan, R.A.Saurous,Y. Agiomyrgiannaki, and Y. Wu, "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), February 2018.

[7] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss†, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. L. Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards end-to-end speech synthesis," Interspeech, April 2017.

[8] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," Conference paper at ICLR, November 2017.

[9] S. Gehrmann, Y. Deng, and A. M. Rush, "Bottom-up abstractive summarization," Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, October 2018.

[10] V. Soni, R. Shaikh, S. Mahato, and S. Phiroj, "T.u.e.s.d.a.y (translation using machine learning from english speech to devanagari automated for you)," International Journal of Computer Applications, June 2021.