T.U.E.S.D.A.Y
(Translation Using machine learning from English Speech to Devanagari Automated for You)

Varun Soni
Don Bosco Institute of Technology
Mumbai,
Maharashtra, India

Rizwan Shaikh
Don Bosco Institute of Technology
Mumbai,
Maharashtra, India

Sayantan Mahato
Don Bosco Institute of Technology
Mumbai,
Maharashtra, India

Shaikh Phiroj
Don Bosco Institute of Technology
Mumbai,
Maharashtra, India

ABSTRACT

In today's globalized world, one thing that acts as a barrier to healthy information exchange is language. On top of that, with the onset of technologies such as YouTube and Facebook making it easy to share knowledge with people all around the world, language can impede that flow of information. If someone in India wants to access a piece of content in written form from another country, they can make use of services such as Google translate. However, the same cannot be extended for any piece of content which is rendered in an audio-visual medium since no such apparatus has been developed which can help people comprehend the content of that specific type. Specifically, students have experienced this first-hand when they try to access content from other universities but the medium of language is something that they are not well-versed in. With these issues in mind, this group is trying to build an automatic voice dubbing system: a speech-to-speech translation pipeline which can help users easily understand other users without the worry of language barrier. The model is called T.U.E.S.D.A.Y. (Translation Using machine learning for English Speech to Devanagari Automated for You) and is divided into three conversion modules: English speech to English text, English text to Devanagari text, and finally Devanagari text to Hindi speech. These modules work together in tandem to ensure an integrated model as a whole.

General Terms

Machine Learning, Deep Learning, Speech Recognition, Machine Translation, Natural Language Processing

Keywords

Neural Networks, TensorFlow, DeepSpeech, Keras, FastSpeech

1. INTRODUCTION

Thanks to the constantly changing world humans are no longer living in cultural silos. Each person from a different culture is interacting with different people from different cultures around the world to learn new perspectives on thinking and doing things. This has been predominantly engineered by social networking websites coupled with video streaming websites.

While more people are interacting through various different means, the methods of interaction havent seen much progress. When it comes to text-based content and messages, the user can make use of services like Google Translate which produces a comprehensible translated piece of content. But when it comes to audio-visual medium, users are not provided with options and mechanisms to comprehend the source language. This gap can hinder a persons quest to learn something new because of the language gap between what the user knows and the original videos language.

To solve this problem the paper proposes an automatic voice-dubbing pipeline system, where the source language will be translated to the target language that the listener wants to listen to. The system will have real-time functionality and will convert and output the audio with minimum lag so that users can comprehend.

Research in this domain has been very limited, hence the system has been divided into three modules. These three modules are speech-to-text, text-to-text, text-to-speech. For time being, only an English-to-Hindi speech pipeline is implemented.

2. LITERATURE REVIEW

The system has been divided into three modules and the literature review is done accordingly.

2.1 Speech to Text

In the research paper titled "Towards Structured deep neural network for automatic speech recognition"[6] written by Yi-Hsiu,

Lin-shan, it is proposed that the Structured Deep Neural Network (structured DNN) be used as a structured and deep learning framework. This approach can learn to find the best-structured object, given a structured input, by globally considering the mapping relationships between the structures rather than item by item.

2.2 Text to Text

There are many available algorithms for text to text translation, but currently, Google is the global leader when it comes to language translation. Google uses a modified version of a neural machine translation system, called Google Neural Machine Translation.

In the paper[8] written by Yonghui, Mike Schuster, Zhifeng, Quoc V. Le, and Mohammad Norouzi, they discuss the strengths of NMTs and their ability to learn directly and mapping from input text to corresponding output text. NMTs have three weaknesses: having a slow training speed, problems with dealing with rare words and being unable to translate all the words accurately all the time.

The GNMT takes care of all these problems. It uses 8 layers LSTM RNNs to encourage gradient flow. To achieve parallelism, the bottom layer of the decoder has an attention unit which is connected to the top layer of the encoder LSTM. Slow training speeds are taken care of by using low-precision arithmetic for inference. Rare words are dealt with using sub-word units. They employ a beam search technique that has a length normalization procedure to take care of the words which do not get translated during decoding.

2.3 Text to Speech

Under this domain, there have been various methods based on concatenative, statistical parametric and end-to-end neural network-based networks. Out of all these, neural networks give the best results but it takes a longer period of time to give output. For a real-time system, this is a major drawback. To tackle this issue of time various models have been proposed taking neural networks as its base architecture and improving upon it by adding further functionalities. Out of the various models Fast Speech was chosen which gives faster results without compromising on speed.

Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhou, Tie-Yan Liu in "Fast Speech: Fast, Robust and Controllable Text to Speech"[7] have proposed a new system that generates Mel-spectrograms in parallel. Generating them in parallel gives faster speeds which are crucial for any real-time application. They have also added functionality called length regulator which matches the source phoneme sequence to Mel- spectrogram sequence. This allows greater control over the generated audio as well as proper audio output from text to audio.

3. PROPOSED ARCHITECTURE

3.1 System Architecture

There are three main modules in the S2ST system: automatic speech recognition (ASR), machine text-to-text (TTT) translation, and text-to-speech (TTS) synthesis.
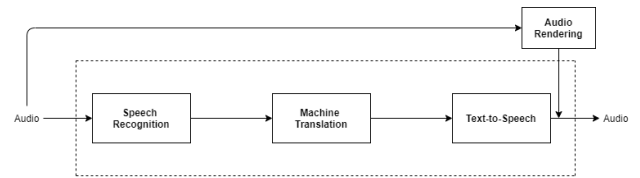

Fig. 1. Entire System Architecture

3.2 Speech to Text

The model has been trained to take an audio sample as input and gives its transcription as an output. As noticed in Fig. 2, the core of the speech-to-text model is a recurrent neural network (RNN). The First 3 being non-recurrent layers. For the very first layer, the output at each time depends on the MFCC (Mel-frequency cepstral coefficients) frame. The other 2 non-recurrent layers function on independent data for each time frame. The hidden units for these 3 non-recurrent layers is a clipped rectified-linear (ReLu) activation function. The fourth layer is a recurrent layer which includes a set of hidden units with forward recurrence. Also, the units at this layer must be computed sequentially. The fifth layer (non-recurrent) takes the forward units as inputs. The output layer is standard logits that correspond to the predicted character probabilities for each time slice and character. Once computed the prediction, CTC loss is computed to measure the errors in prediction. Figure 2 shows the complete RNN Deep-Speech[1] model.
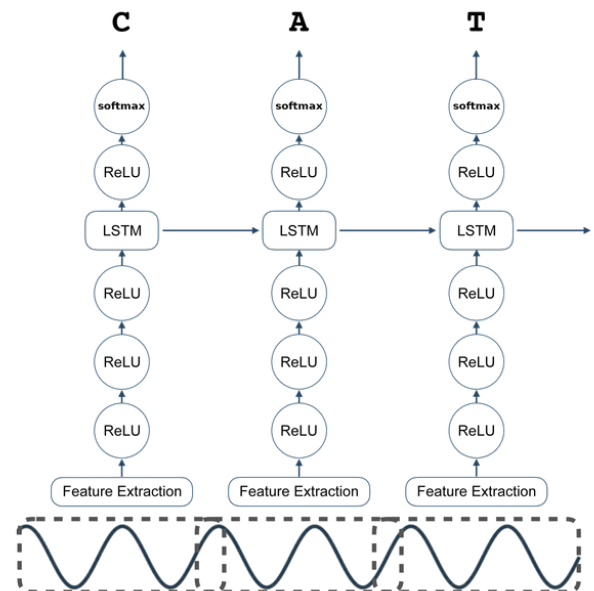

Fig. 2. Deep-speech RNN Model

3.3 Text to Text

The figure given below is the model architecture for text-to-text machine translation. The model is a neural network made up of 7 layers: two input layers, two embedding layers, two LSTMs and finally a dense layer.
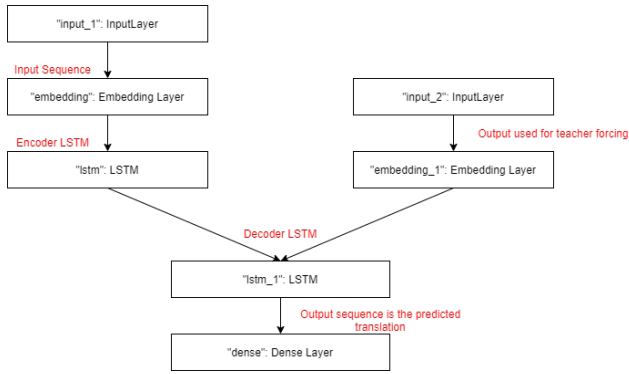
Fig. 3. Text-to-Text Model Architecture

The first input layer provides the input sequence to the first embedding layer. The other input layer is used for teacher forcing and its output is supplied to the second embedding layer. The first embedding layer is connected to the LSTM layer, which acts as an encoder. This encoder LSTMs output will be a vector of the size which is the same as the vocabulary size, representing the probability distribution of each word in that vocabulary. The second LSTM acts as the decoder LSTM, whose initial states are the final states of the encoder. For teacher enforcing, the actual translation is supplied to the second input layer. The overall loss is calculated on the predicted output, given by the final Dense Layer. The dense layer calls the decoder LSTM in a loop and generates one word at a time. The loop breaks when the machine predicts an _END token. So the model has two outputs, the actual Hindi translation and the predicted output sequence (dense layers output).

3.4 Text to Speech

For the text to speech module, a transformer based model is used with an overlay of feed-forward network which helps to generate mel-spectrograms in parallel. This increases the overall efficiency and speed of generating audio output from the text.

(a) Feed-Forward Transformer (b) FFT Block (c) Length Regulator (d) Duration Predictor
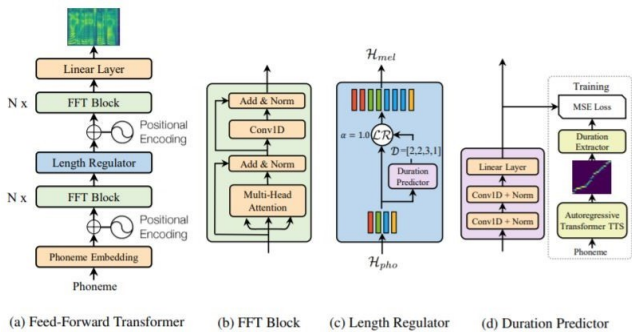
Fig. 4. Text-to-Speech Model Architecture

This Feed-forward transformer consists of blocks which contain a self-attention and 2-layer 1D convolution network with Relu activation function. Multiples of these FFT blocks are placed on both phoneme and mel-spectrogram sides.

To make the outputs streamline the model uses a length regulator and duration prediction which helps to map each phoneme to its generated mel-spectrograms. This mapping helps in generating accurate outputs as well as provides more control to the user

4. IMPLEMENTATION

4.1 Speech to Text

With the help of a deep-speech model and a punctuator, an accuracy of 5.6% WER is achieved on Librivox clean test.
The extracted audio from the video is fed to a deep speech model which is mainly composed of an Acoustic model and decoder. The decoder uses a beam search algorithm to output textual transcript. The output is then passed to a punctuator model which punctuates the transcript. The final output can be a JSON file with timestamps and also a regular text file. For compatibility, the text file is passed to the next module.

4.2 Text to Text Translation

The working of Encoder-Decoder Structure in the NMT:

Fig. 5. Encoder-Decoder Structure

Using Tensorflow [5], specifically Keras [3], the encoder is fed each word of the input sentence periodically. After every time period (t), the model will update a hidden vector (h) which keeps the information about the input word, while also keeping the old information from previous time periods. Some weight matrices are also fed to the machine to improve the accuracy while training. At the final step, the hidden vector produced as output is called the thought vector and is fed to the decoder as an input. The decoder gets a $<$SOS$>$ tag as the first input, after which it updates the time- period to t=1. It uses an additional weight matrix to create a probability of all the words in the output vocabulary. Hence, the word with the highest probability becomes the first word of the sentence.

```
latent_dim = 300

# Encoder
encoder_inputs = Input(shape=(None,))
enc_emb = Embedding(num_encoder_tokens+1, latent_dim, mask_zero = True)(encoder_inputs)
encoder_lstm = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder_lstm(enc_emb)

# Discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]
```

Fig. 6. Encoder Algorithm

```
# Decoder
decoder_inputs = Input(shape=(None,))
dec_emb_layer = Embedding(num_decoder_tokens+1, latent_dim, mask_zero = True)
dec_emb = dec_emb_layer(decoder_inputs)

# Decoder returns full output sequences and internal states as well.
# Return states not used in the training model, but later in inference.
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(dec_emb,
                                    initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
```

Fig. 7.   Decoder Algorithm

4.3   Text to Speech

The FastSpeech model has been implemented using Python as the base language. The model has been trained on LJ Speech dataset which comprises of more than 13,000+ short audio clips. The model is able to generate English speech successfully , but unfortunately, it is not able to generate Hindi speech for which it was proposed. Further research needs can help rectify its working.

5.   RESULTS

The system can translate any video which has clear speech and limited vocabulary. Below is the result obtained from the system when an input video is passed. The actual transcription from the video is as follows:
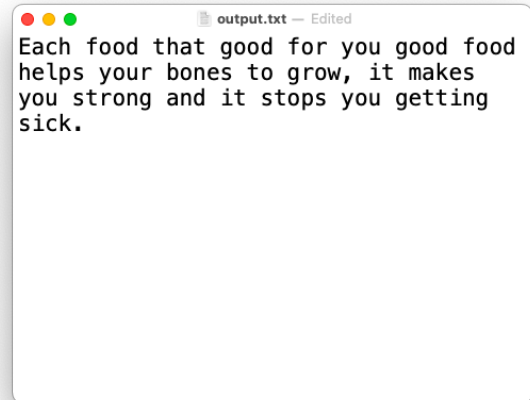
> 'Eat food that's good for you good food helps your bones to grow, it makes you strong and it stops you getting sick.'

First, the code is run, then the video to be translated it selected and the model runs inference:
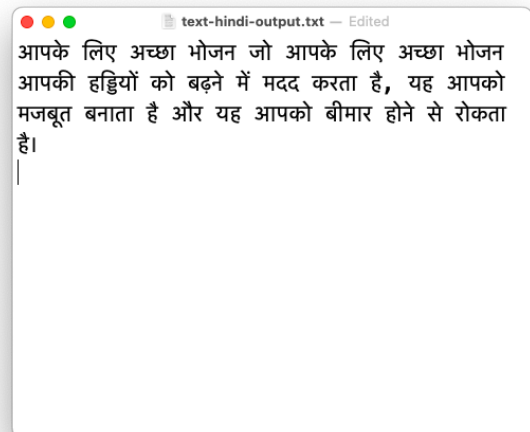

Fig. 8.   Running the code

After this, we get two output text files, 'output.txt' has the English transcripts and 'text-hindi-output.txt' has the Devanagari translation from the English output file.

Fig. 9.   Text converted to English

Fig. 10.   Hindi translation of the English output

As one can notice, the translation results are not 100% accurate, with the model mistaking the word 'eat' for 'each'. However, other than some small mistakes, the model is pretty accurate, with sentences which are, for the most of it, semantically correct.
Similarly, for another test case, these were the output text files:

```
[ ]  import torch
     waveglow = torch.hub.load('nvidia/DeepLearningExamples:torchhub', 'nvidia_waveglow')

     Downloading: "https://github.com/nvidia/DeepLearningExamples/archive/torchhub.zip" to /root/.cache/torch/hub/torchhub.zip
     Downloading checkpoint from https://api.ngc.nvidia.com/v2/models/nvidia/waveglowpyt_fp32/versions/1/files/nvidia_waveglowpyt_fp32_20190306.pth

[ ]  waveglow = waveglow.remove_weightnorm(waveglow)
     waveglow = waveglow.to('cuda')
     waveglow.eval()
     mel = mel.cuda()

[ ]  # run the models
     with torch.no_grad():
         audio = waveglow.infer(mel)
     audio_numpy = audio[0].data.cpu().numpy()

[ ]  plt.figure(figsize=(14, 5))
     librosa.display.waveplot(audio_numpy, sr=22050)

     <matplotlib.collections.PolyCollection at 0x7f4300069e10>

[ ]  ipd.Audio(audio_numpy, rate=22050)

     ▶  0:00 / 0:08 ──────────  🔊  ⋮
```

Fig. 13. Successful English speech output in Colab
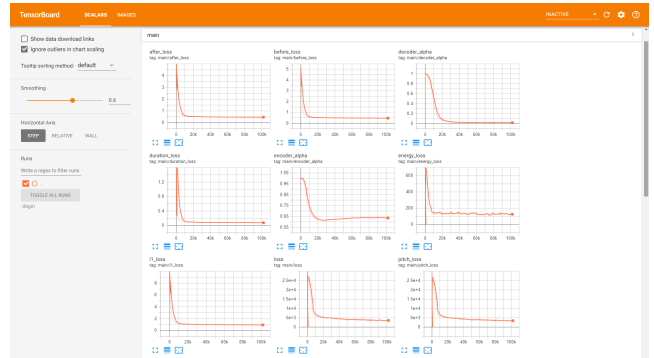
Fig. 14. Tensorboard graphs displaying the various parameters related to training

6. CONCLUSION AND FUTURE WORK

We were able to make individual models to carry out respective translation tasks. The results are semantically correct, with some grammatical errors. This project let us learn about natural language processing and the different algorithms involved in machine translation specifically. We also learnt how to handle, clean, and normalise data sets in order to use them for the respective models. However, the modules are yet to be integrated and implemented as a whole.

Coming to the future scope, firstly, the system has yet to integrate all the models together successfully so as to have an entire system translating videos from English to Hindi. Next, to increase the accuracy, larger data sets need to be found and used. The system can be extended to be used with a broad set of vocabulary and noisy speech. Furthermore, support for different languages can be added.

7. REFERENCES

[1] Deepspeech - an open-source speech-to-text engine. `https://github.com/mozilla/DeepSpeech`. Accessed: 2020-05-13.

[2] Google colab - a python development environment that runs in the browser using google cloud. `https://research.google.com/colaboratory/`. Accessed: 2020-05-13.

[3] Keras - an open-source software library for artificial neural networks. `https://keras.io/`. Accessed: 2020-05-13.
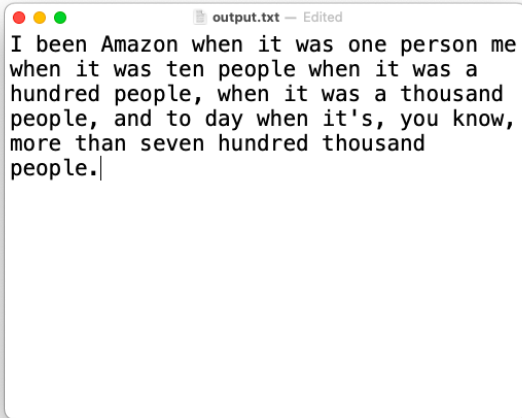
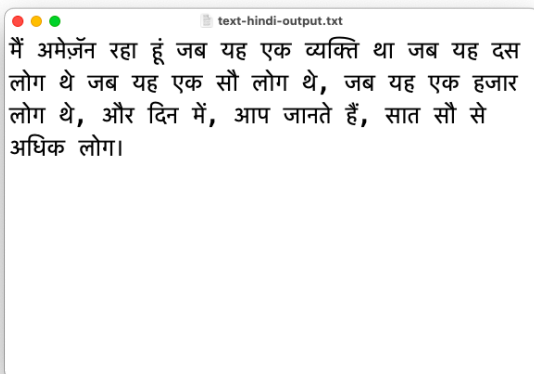Fig. 11. Text converted to English

Fig. 12. Hindi translation of the English output

As mentioned before, the model could not generate Hindi speech but it could generate English speech sounds. The following screenshots from Google Colab[2] and Tensorboard shows its successful working.
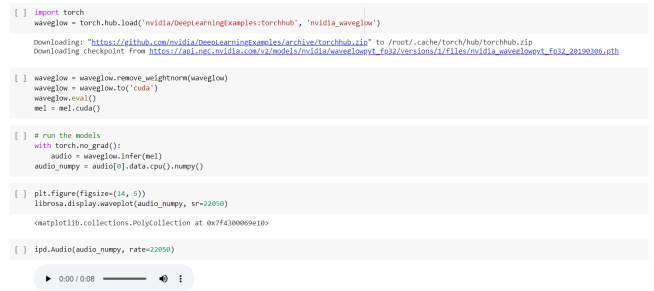
[4] Tensorboard - tensorflow's visualization toolkit. `https://www.tensorflow.org/tensorboard`. Accessed: 2020-05-13.

[5] Tensorflow - a free and open-source software library for machine learning. `https://www.tensorflow.org/`. Accessed: 2020-05-13.

[6] Yi-Hsiu Liao, Hung-yi Lee, and Lin-shan Lee. Towards structured deep neural network for automatic speech recognition. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 137–144. IEEE, 2015.

[7] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech, 2019.

[8] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.