Text to Speech Generation for Regional Languages

Problem Statement

The recent advancements in the speech processing domain are speech recognition, speech synthesis, speech analysis and coding. Speech synthesis is the artificial production of human speech. A computer system used for the purpose is called a speech synthesizer and can be implemented in software or hardware products. A text-to-speech (TTS) system converts normal language text into speech; this problem focuses on TTS for regional languages. TTS applications are well known as assistive aides for people who experience dyslexia, reading challenges, or visual impairment. The other valuable uses for the application range from reducing eye strain from reading (digital or paper formats), reducing paper use due to printing digital text, foreign language learning, writing and editing, or promoting listening skills.

Background

The initial work was done in MATLAB environment for TTS generation, one of the paper discuses Kannada language TTS generation using MATLAB. Many different Machine learning and deep learning approaches have been seen in recent years. WaveNet is a powerful generative model of audio. It works well for TTS but is slow due to its sample-level autoregressive nature. It also requires conditioning on linguistic features from an existing TTS frontend, and thus is not end-to-end: it only replaces the vocoder and acoustic model. Another recently-developed in 2017 in neural model is DeepVoice, which replaces every component in a typical TTS pipeline by a corresponding neural network. However, each component is independently trained, and it's nontrivial to change the system to train in an end-to-end fashion.

In year 2016, the earliest work touching end-to-end TTS was using seq2seq with attention. However, it requires a pre-trained hidden Markov model (HMM) aligner to help the seq2seq model learn the alignment. It's hard to tell how much alignment is learned by the seq2seq per se. Second, a few tricks are used to get the model trained, which the authors note hurts prosody. Third, it predicts vocoder parameters hence needs a vocoder. Furthermore, the model is trained on phoneme inputs and the experimental results seem to be somewhat limited. Char2Wav (2017) is an independently-developed end-to-end model that can be trained on characters. However, Char2Wav still predicts vocoder parameters before using a Sample RNN neural vocoder (2016), whereas Tacotron directly predicts raw spectrogram. Also, their seq2seq and Sample RNN models need to be separately pre-trained, but our model can be trained from scratch.

Methodology

We will be discussing a method that was proposed by Arun et. al. In 2014. The model is based on Convolutional Neural Network(CNN) with combination of Gaussian mixture model - hidden Markov model (GMM-HMM). The following block diagram gives the insight of working of the model.

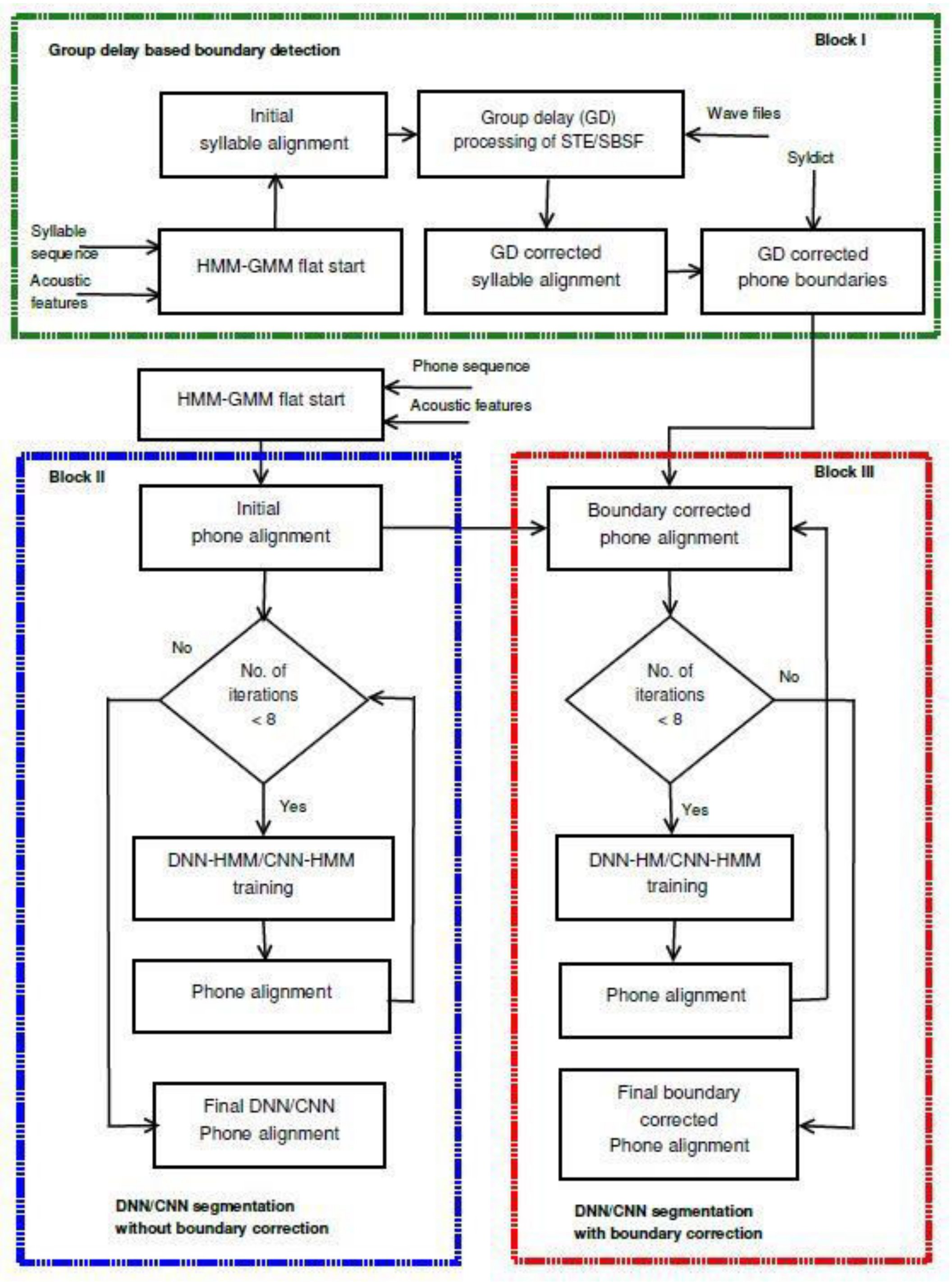
Group delay based boundary detection
Block I
Initial syllable alignment
Group delay (GD) processing of STE/SBSF
Wave files
Syldict
Syllable sequence
Acoustic features
HMM-GMM flat start
GD corrected syllable alignment
GD corrected phone boundaries
HMM-GMM flat start
Phone sequence
Acoustic features
Block II
Initial phone alignment
No
No. of iterations < 8
Yes
DNN-HMM/CNN-HMM training
Phone alignment
Final DNN/CNN Phone alignment
DNN/CNN segmentation without boundary correction
Block III
Boundary corrected phone alignment
No. of iterations < 8
No
Yes
DNN-HM/CNN-HMM training
Phone alignment
Final boundary corrected Phone alignment
DNN/CNN segmentation with boundary correction

Figure 1: Block Diagram of Method ["Deep Learning Techniques in Tandem with Signal Processing Cues for Phonetic Segmentation for Text to Speech Synthesis in Indian Languages" by Arun Baby, Jeena J Prakash, Rupak Vignesh, Hema A Murthy. In INTERSPEECH 2017]

Neural networks are not used for speech segmentation in the TTS framework for Indian languages even though they are widely used in speech recognition. In this work, GMMs in HMM-GMM framework for phoneme segmentation in TTS systems are replaced by DNN and CNN for better phoneme segmentation. Acoustic models are built by training the neural networks with the GMM-HMM monophone alignment (also known as HMM-based phone alignment) as the initial alignment. The DNN-HMM/CNN-HMM are then trained iteratively to get accurate final phone boundaries. This is shown in Block II of Figure 1. The number of iterations is set to 8 empirically as the phone boundaries do not change much afterward.

Acoustic cues give robust syllable boundaries for a subset of syllables. Syllable boundary correction using signal processing cues (GD of STE, and SBSF) after GMM-HMM flat start initialization is shown in Block I of Figure 1. The boundaries of the last phone of each corrected syllables are marked as GD corrected phone boundaries. A syllable to phone dictionary (shown as syldict in Figure 1) is used to map from syllable to phoneme sequence. Most of the phone boundaries given by neural networks are better than GMM-BC alignment, which uses signal processing cues along with GMMHMM based forced alignment.

The framework, where the boundaries obtained using DNNs/CNNs are further corrected using signal processing cues is shown in Block III of Figure 1. Similar to segmentation using deep networks, GMM-HMM monophone alignment is used as the initial phone alignment. These phone alignments are corrected, either forward or backward, using GD corrected phone boundaries. The boundary corrected phone alignments are then used for training neural networks. The alignments obtained after deep network training are again corrected using GD corrected phone boundaries and this process is repeated 8 times iteratively. After the 8th iteration, phone alignments obtained from deep networks are corrected again using GD corrected phone boundaries as shown in Figure 1.

Experimental Design

Dataset: There are very few datasets available for Indian languages. The dataset can be found on this link.
Evaluation Measures: The model can be evaluated by using word error rate and degradation mean opinion score.
Software and Hardware Requirements: Python based Deep Learning libraries will be exploited for the development and experimentation of the project. Tools such as Anaconda Python, and libraries such as Tensorflow, and Keras will be utilized for this process.