Automatic Speech Recognition Framework for Indian Languages

Shubhangi Ghosh

Indian Institute of Technology Madras

Abstract. In this project, we worked on developing several automatic speech recognition models for Indian languages, namely Tamil, Telugu and Gujarati using the Kaldi Speech Recognition Toolkit. A HMM-GMM acoustic model in conjugation with a n-gram language model was initially built for converting speech in the above mentioned languages to text. To obtain improved Word Error Rates, a Time Delay Neural Network (TDNN) was run. A Recurrent Neural Network based Language Model (RNNLM) pipeline was then set up to improve the contextual information compared to the n-gram language model. To achieve End-To-End speech recognition, CTC (Connectionist Temporal Classification) was used in conjugation with an Encoder-Decoder framework. A detailed ananlysis of this framework was performed to obtain best results.

1 Introduction

Due to the large diversity of languages in India, an efficient and feasible implementation of an Automatic Speech Recognition framework is an absolute necessity. This would enable large scale collaboration between urban dwellers and communal workers and make for a profitable work environment for all. We have exploited conventional speech recognition methods such as the HMM-GMM framework along with modern Deep Learning based frameworks to achieve the same. A RNNLM pipeline was also set up to improve word (language model) priors. Various modules from the Kaldi Speech Recognition Toolkit were used to achieve the above. An ESPNET module was used to achieve End-To-End Speech Recognition. A detailed analysis of this model was performed and the framework was tuned to obtain best results.

2 Conventional HMM-GMM framework

2.1 Acoustic Model

The acoustic model uses a phone alignment framework offered by KALDI. Mel Frequency Cepstral Coefficient (MFCC) based features are extracted for each frame. Each frame is represented as a sequence of three states. A GMM-HMM model is trained, optimized using Viterbi algorithm to maximize phone (thereby, frame) alignment to output text.[1]

2.2 Language Model

N-gram language models are used. Unigram, Bigram and Trigram language models were experimented with in our case.

2.3 Feature Extraction

Feature representations are improved using feature space transforms like the LDA and adding delta and delta-delta features (dynamic speech representation).

3 Time Delay Neural Network

Time Delay Neural Networks are used instead of BLSTMs are BLSTMs can get bulky. TDNN is essentially a feed-forward neural network where the output is fed explicitly at a delayed input time to make the network more lightweight. Temporal convolution is used to obtain forward context. [2] BLSTMs usually operate at a speed of 33 frames per second while TDNNs can process 100 frames per second.
Fbank features are used as input. Posterior probabilities of phone to frame alignment obtained from n-gram models are used as targets by the TDNN.

4 Results

The following best results were obtained on running the above model on Telugu data obtained from Microsoft Interspeech challenge.

–	tri1	tri2	DNN
WER	35.16	34.03	29.32

Model	Tied States	No of GMM Mixtures
tri1	2000	16
tri2	1600	20

The hyperparameter setting is described as follows:
4 hidden layers, 1024 neurons per hidden layer, Gradient Descent optimiser, Initial Learning rate: 0.008, Loss function: Cross entropy

5 RNNLM

5.1 Data Collection and Cleaning

Data Collection

- Data was collected from Wikimedia XML dumps of text from all Wikipedia pages existent in a given language. Here are the corresponding links for Tamil, Telugu and Gujarati.

- The dump was extracted using a Github module called Wikiextractor.

Data Cleaning

- Only words which were entirely in the native language were retained. This was done by finding the maximum and minimum hexadecimal value of characters in every given. The word was retained in the cleaned data only if both the maximum and minimum value lie in the hexadecimal range of the given language.
- Sentences were placed in separate lines by programmatically adding a new line after every full stop. Between two files five dummy words were placed to have context gaps.

5.2 Training

Training was done using the train script from Kaldi RNNLM module. The same vocabulary set was used as used for training n-gram models as the final word lattice obtained from n-gram model is used for re-scoring with RNNLM probability values.

5.3 Decoding

RNNLM assumes infinite context for word-based language model. Hence it cannot be used for directly generating a word lattice and decoding word probabilities. Hence decoding is done by re-scoring the lattice obtained from a n-gram language model.[4]

6 End-To-End Speech Recognition

6.1 Connectionist Temporal Classification

This is a character-based model for speech recognition. Every input frame is mapped to a character belonging to the alphabet of the language or the null symbol. Multiple consecutive frames may be mapped to the same character. Thus output sequence lengths may be variable which is our main motive to use CTC. However output length is upper-bounded by input length. Character predictions take place using a Recurrent Neural Network (RNN). The objective is to maximise the probability of the output given the input. A more detailed description can be found here.

The CTC model assumes level-1 Markov assumption of conditional independence. This means that the predicted character(from alphabet or null symbol) for the next frame depends only on the predicted character for the current frame.

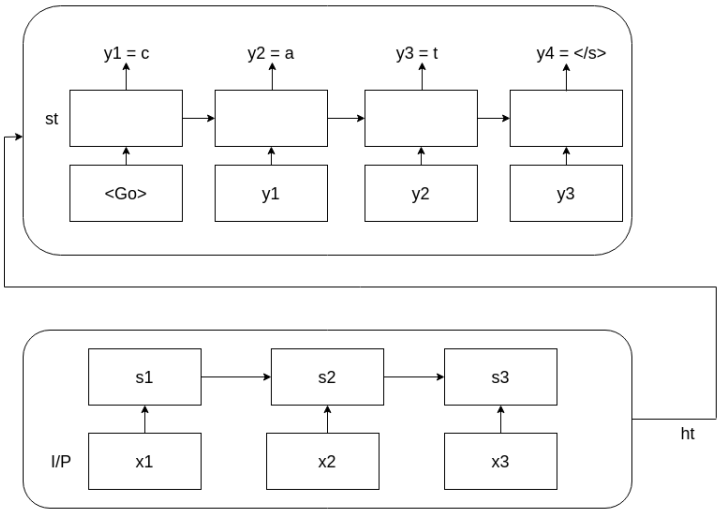
6.2 Encoder-Decoder Model

The drawbacks of the CTC model that the Encoder-Decoder model tries to overcome are as follows. It does not upper bound the output sequence length to the input sequence length. Also, it does not make any conditional independence assumption and assumes infinite context.

$$p(C|X) = \Pi_l P(c_l|c_1, ...c_{l-1}, X)$$

Encoder

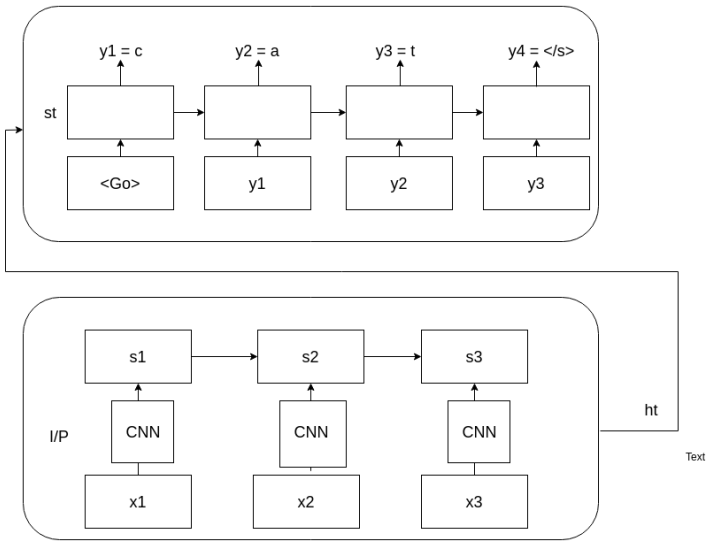
Fig. 1. BLSTM Encoder - LSTM Decoder



This model compensates too flexible alignment properties in the attention-based method with CTC as a regularization during training and as a score correction during decoding.
The encoder encodes the speech frames into state representation and subsequently produces an output at every time-step. There are two choices of encoder in our model. One used a simple Bidirectional LSTM (BLSTM). The other uses a Convolutional Neural Network (in our case, VGGNET) abstraction of the speech frames and runs the BLSTM on those representations. This is denoted as $Encoder(X) = BLSTM(X)$ or $BLSTM(VGGNET(X))$.

Attention Mechanism
At certain time-steps, the output is dependent only on certain frames in the input. Thus an attention mechanism is incorporated to assign appropriate attention weights to corresponding frames in the input. Attention weights range from

Fig. 2. BLSTM(CNN) Encoder - LSTM Decoder



0 to 1 and sum up to 1, and are computed as a function of the decoder state and corresponding encoder outputs. We denote this function as $Attention(h_t, s_{l-1}))$

Decoder
At every decoder time step t, the attended encoder output, previous state and previous time step's decoder output is fed to the decoder unit. the Decoder is, $Decoder(X) = Softmax(Lin(BLSTM(.))$. The Decoder output sequence can only be of a fixed length.
The ESPNET toolkit was used to implement End-To-End speech recognition[1]. More details about the implementation of End-to-End Speech Recognition can be found here.[3]

$$h_t = Encoder(X)$$

$$a_{lt} = Attention(h_t, s_{l-1})$$

$$r_l = \Sigma_t a_{lt} h_t$$

$$p(c_l|c_1, ...c_{l-1}) = Decoder(r_l, s_{l-1}, c_{l-1})$$

how is it combining to words

6.3 Results:

Now, we present the Word Error Rates (WERs) of the above End-to-End Speech Recognition Model on three languages. All results presented are %WERs.

[1] I would like to acknowledge Anirudth N. for his extensively working with me to figure out the end-to-end speech recognition pipeline.

Best Results:

	Tamil	Telugu	Gujarati
Best WER	36.4	21.8	19.1

Hyperparameter Settings for best results
The Hyperparameter settings that gave the best results for all three languages is described as follows.

Batch size: 30
Epochs: 15

Encoder:
Encoder(X) = BLSTM(VGGNET(X))
No. of Layers: 8(Tamil), 4(Telugu, Gujarati)
Encoder Units: 320(Tamil), 160(Telugu, Gujarati)
Encoder Projection Length: 640

Decoder:
No. of Layers: 4
Decoder Units: 300

BLSTM vs. VGGBLSTM Encoder

	Tamil	Telugu	Gujarati
BLSTM	42.6	24.3	21.6
VGGBLSTM	40.1	23.2	20.1

We see an improvement in WER using VGGBLSTM as VGGNET obtains a more meaningful feature abstraction than the standard fbank features.

Encoder Layers

Encoder Layers	Tamil	Telugu	Gujarati
4	40.1	23.2	20.1
8	38.7	23.7	21.5

We see that increasing number of Encoder layers results in an improved WER only for Tamil. This is because Tamil has a significantly larger character set than Telugu or Gujarati and using higher level abstractions makes more sense for Tamil. For Telugu and Gujarati, the search space for optimization is increased and we don't end up achieving optimal results.

Encoder Units

Encoder Units	Tamil	Telugu	Gujarati
160	39.9	22.8	19.8
320	38.7	23.2	20.1
640	40.7	24.6	22.5

Increasing encoder units basically increases the context size for frames. Again this results in an improved WER only for Tamil.

Encoder Projection Size

Encoder Projection Size	Tamil	Telugu	Gujarati
320	38.7	22.8	19.8
640	37.6	22.4	19.6

Increasing Encoder Projection size results in a WER improvement because it allows for a more sparse representation of features.

Decoder Layers

Decoder Layers	Tamil	Telugu	Gujarati
2	37.6	22.4	19.6
4	36.4	21.8	19.1

Increasing number of layers in decoder results in a WER improvement because it allows higher level abstraction in Decoder.

References

[1] Mark Gales, Steve Young, et al. "The application of hidden Markov models in speech recognition". In: *Foundations and Trends® in Signal Processing* 1.3 (2008), pp. 195–304.

[2] Vijayaditya Peddinti et al. "Low latency acoustic modeling using temporal convolution and LSTMs". In: *IEEE Signal Processing Letters* 25.3 (2018), pp. 373–377.

[3] Shinji Watanabe, Takaaki Hori, and John R Hershey. "Language independent end-to-end architecture for joint language identification and speech recognition". In: *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*. IEEE. 2017, pp. 265–271.

[4] Hainan Xu et al. "A Pruned RNNLM Lattice-Rescoring Algorithm for Automatic Speech Recognition". In: (2018).