CS230

HMM-Viterbi Based Decoders vs Deep Neural Architectures (Speech Recognition)

Erol Aygar*
Department of Computer Science
Stanford University
aygar@stanford.edu

Abstract

ASR technology is undergoing architectural changes recently. Hidden Markov Models (HMMs) are a mainstream statistical graphical representation for modeling vector sequences in the temporal domain [Koller]. Deep Neural Networks (DNNs) with time sequence capabilities (RNN), and a class of new recognizers (E2E NNS), as well as hybrid models, offer alternatives for current design decisions [Nassif] [Liu]. Novel learning and representation methods enabled better language and acoustic models. Large datasets, open-source toolkits, and computational infrastructures are publicly available. Accuracy of the state-of-the-art decoders improved from 16 to 6 percent accuracy in large vocabulary corpus in the last decade [cs224s]. This paper proposes a study on comparing HMM based recognizers with Deep Learning based methods.

1 Introduction

1.1 Learning Methods

The primary goal of learning in speech is to build a model to infer the text sequences from a sequence of feature vectors. The features are used both in training and recognition processes. The following figure shows inputs and outputs of training and recognition processes.

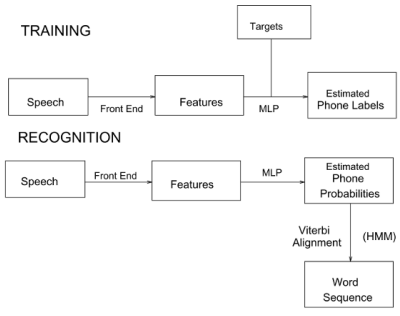

Figure1-Training and recognition processes [Renals].

*https://www.linkedin.com/in/erolaygar/

1.2 Conventional Methods

CMU-Sphinx is an example of open source HMM-based ASR toolkits. The high level design of its decoding process is laid out on the following figure [Lamere]. The goal is to study a Deep Learning alternative that will be comparable with this design. CMU-Sphinx framework has three main design blocks: Frontend (end pointer, feature computations), Decoder (graph construction, state probability computation, search), and Knowledgebase (dictionary/lexicon, language model, acoustic model).

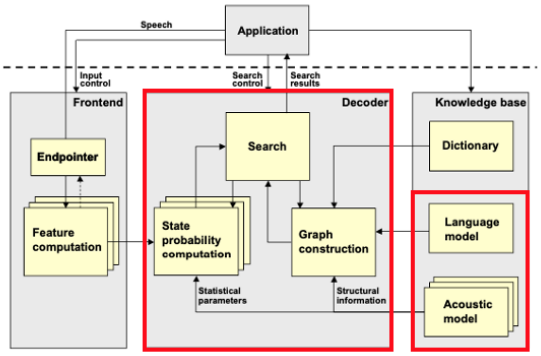

Figure2-Architecture of the Sphinx-4 system [Lamere]. Red marks our focus.

Frontend takes in speech signals and generates feature vectors. The decoder block performs the actual recognition. The graph construction module translates any type of standard language model into an internal format. It constructs a Language HMM from the phonetic dictionary and structural information from one or more sets of acoustic models (Gaussian mixture). The graph construction module has two submodules. The first interprets the language model and converts it into an internal grammar. LM can be in multiple forms (e.g., statistical N-grams, context-free grammars) Internal grammar is then converted to a language HMM. Language HMM was then used by the search module to determine the structure of the trellis to be searched. The search is performed by Viterbi or Bushderby algorithms. In HMM-based decoders, search is performed through a trellis (a directed acyclic graph or DAG which is the cross product of the language HMM and time). The state probability computation module has access to the feature vectors (39 MFCC) and computes the state output probabilities from features.

1.3 Motivation

The motivation of this project is to understand the conventional as well as the novel Deep Learning-based ASR model architectures to have a baseline for future studies. Our goal is to implement a Deep Learning Decoder and compare its performance with the existing ones based on training and decoding accuracy.

2 Model Design

After completing the course modules and meeting with project mentoring TAs, instead of implementing a network solution from scratch, we decided on RNN that use Long Short Term Memory (LSTM) Cells as our sequence model architecture and DeepSpeech Framework as the started code for the implementation. The framework uses a Connectionist Temporal Classification [CTC] loss function [DeepSpeechM]. The framework implements a Recurrent Neural Network (RNN) that can be trained to input speech spectrograms or audio files to generate transcriptions in text format and The code lets the application programmer make changes to its code and adjust the hyperparameters for the experiments as well as let use external language models to measure and improve the accuracy of the predictions. It supports beam search as a decoding algorithm by default with possible support for external alternatives. The input audio files are converted into a power spectrum of signals during a feature extraction preprocessing step. They are also called features in short or Mel-frequency cepstral coefficients (MFCC's), which are also commonly used input formats for HMM-based acoustic models.

3 Model Explanation

The model can be summarized as following formulation [DeepSpeech]. Let single utterance be denoted as x and the label y be sampled from a training set.

$$S = \{x^{(1)}, y^{(1)}, x^{(2)}, y^{(2)}, ...\}$$

Let utterances, $x^{(i)}$ be time-series of length $T^{(i)}$ where every time-slice is a vector of audio features, $x_t^{(i)}$ where $t = 1, ..., T^{(i)}$ MFCC's are used as the features; so $x_{t,p}^{(i)}$ denotes the p-th MFCC feature in the audio frame at time t. The goal of the RNN is to convert an input sequence x into a sequence of character probabilities for the transcription y, with $\hat{y}\ t = \mathbb{P}c_t|x$, where for the alphabet $c_t \in \{a, b, c, ..., z, space, apostrophe, blank\}$ The RNN model has 5 layers of hidden units. For an input x, the hidden units at layer l are denoted hl and hl is the input layer. The first three layers are not recurrent layers. The first layer, at each time t , the output depends on the MFCC frame x_t along with the context of C frames on each side. The remaining non-recurrent layers operate on independent data for each time step. Thus, for each time t, the first 3 layers are computed by:

$$h_t^{(l)} = g(W^{(l)}h_t^{(l-1)} + b^{(l)})$$

where $g(z) = min(max(0, z), 20)$ is a ReLu activation function and $W^{(l)}, b^{(l)}$ are the weight matrix and bias parameters for layer l The fourth layer is a recurrent layer and includes a set of hidden units with forward recurrence $h^{(f)}$

$$h_t^{(f)} = g(W^{(4)}h_t^{(3)} + W^{(f)}h_{t-1}^{(f)} + b^{(4)})$$

$h^{(f)}$ is computed sequentially from $t = 1$ to $t = T^{(i)}$ for the -th utterance. The fifth layer is non-recurrent and takes the forward units as inputs.

$$h^{(5)} = g(W^{(5)}h^{(f)} + b^{(5)})$$

The output layer is standard logits that correspond to the predicted character probabilities for each time slice t and character k in the alphabet:

$$h_{t,k}^{(6)} = \hat{y_{t,k}} = (W^{(6)}h_t^{(5)})_k + b_k^{(6)}$$

Where $b_k^{(6)}$denotes the k-th bias and $(W^{(6)}h_t^{(5)})_k$ the k-th element of the matrix product. The CTC loss $\mathcal{L}(\hat{y}, y)$ is computed after a prediction for $\hat{y_{t,k}}$ is computed. The loss function uses the *blank* symbol for transitions between characters. During training, the gradients $\mathcal{L}(\hat{y}, y)$ with respect to the network outputs are calculated based on the ground-truth character sequence y and the gradients with respect to model parameters are applied by the back-propagation function.

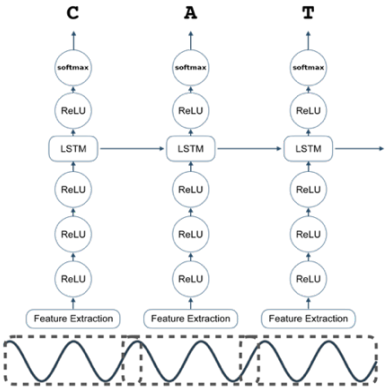

Figure3-LSTM-RNN-CTC Model Architecture [DeepSpeech].

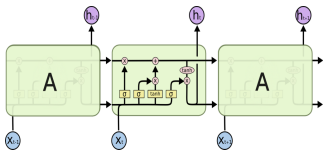

Figure4-The RNN consists of an LSTM RNN that works "forward in time" [DeepSpeech].

4 Dataset and Features

The target datasets consist of four corpora in different characteristics. TIDigits, TIMIT, Switchboard, and NOAA Corpus. The idea is to benefit from different features of each dataset, such as vocabulary size, the structure of conversations, speaker gender, and accent variations. The following list explains the corpora used.

Switchboard (Large) is a collection of spontaneous conversations among 543 speakers (302 males, 241 females) from every major dialect of American English. The corpus amount to 259 hours of recorded speech, and about 3 million words of text. contains 2438 two-sided telephone conversations stored in two-channel audio files averaging 6 minutes in length. Audio files are two channels, 8khs each, NIST Sphere formatted files. Each audio file has a transcriptions text file that contains the word alignments. The switchboard dataset with Gaussian Mixture Models has a benchmark WER of %25.2 as of 2015.

TIMIT (Mid Size) contains recordings of 630 speakers of eight major dialects of American English, each reading ten phonetically rich sentences. The corpus includes 16kHz audio files for each utterance and time-aligned word transcriptions.

NOAA [Small] is recordings and manually transcription of radio weather messages broadcast by the National Oceanic and Atmospheric Administration (NOAA). The dataset included 185 total audio files. The transcriptions and word alignments are done manually.

TIDigits (Tiny) There are 326 speakers (111 men, 114 women, 50 boys, and 51 girls) each pronouncing 77-digit sequences. Dataset is partitioned into test and training subsets by default.

5 Model Training

5.1 Data Preperation

We implemented python scripts to format the datasets into DeepSpeech's expected format. Each dataset is split into test, development and test sets. The data size is larger so we segmented into subsets of each dataset for development and discovery purposes. Also, extracting data into wav files and split them to utterances based on the time slices provided in the transcriptions. This phase was the most time consuming phase of our experiments along with the computation time during training.

5.2 Parameters

Table 1 shows the inputs and hyperparameters that are used to tune the model and it's capabilities. Third column shows the list of values we came up with during the study and used to train the LSTM-RNN-CTC model.

Parameter	Description	LSTM-RNN-CTC
n-hidden	number of hidden layers	2048
early-stop	early stop enabled	True
es-min-delta	limit used for early stopping	0.1
es-epoch	n epocs without expected change on loss	6
epochs	number of epochs	200
dropout-rate	dropout rate	0.2
learning-rate	learning rate	0.00095
training-batch-size	training batch size	80
dev-batch-size	development batch size	80
test-batch-size	test batch size	40
alphabet-config	alphabet file	alphabet.timit
lm	n-gram language model binary	lm.bin
trie	trie for language model	lm.trie

Table 1 -Model inputs and hyperparameters

We ran 200 epochs on TIMIT dataset with the hyperparameters. Overfitting is observerd after 55-60 epocs. %37 WER observed on the development set. Figure shows the learning curve of the training process.

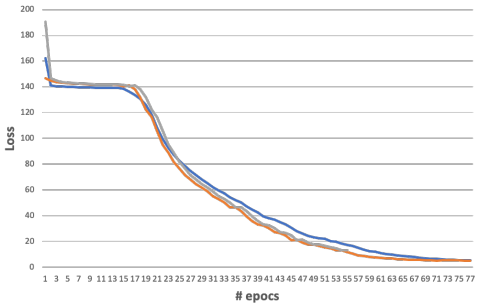

Figure5 - Learning Curve

6 Evaluation

6.1 Experiment Design

The experiment compares recognizers and report results based on their Accuracy (ACC: (N-D-S)/N), Word Error Rate (WER: I + D + S), Speed, and Implementation complexity. The experiment will train both recognizers with the four datasets mentioned in related the section above. We will segment each corpus into sub dataset sizes (tiny, small, med, large, x-large, full) and compare recognizer performances based on unseen and test on train data. Both solutions will use the same Dictionary (CMU Dictionary), Language Model (n-gram).

6.2 Scoring

Scoring is the process of rating the quality of the models created. We use SCLITE to generate the scores to compare the decoding performances of the models [SCLITE]. The process compares two transcripts. SCLITE compares each line of the hypothesis transcript with the reference transcript, counting the number of times a word is substituted for another word is added or deleted.

6.3 Experiment Results

We partially could complete all experiment runs and use the SCLITE scores to report the accuracy of our models instead. HMM (Sphinx) have reached %41.8 WER on unseen data as the best result on 300-hour train and decode on full set of Switchboard dataset. LSTM-CTC based on RNN Cells achieved %55.4 WER on test data and %64.8 WER on unseen data (switchboard) with limited fine tuning and experience. Although we had limited time and experience on finetuning the model parameters and transfer learning, we have achieved %20.4 WER on test set of TIMIT corpus based on the pre-trained model (Switchboard, Fisher, LibriSpeech, Switchboard, Common Voice English)

Dataset	HMM-BW-Viterbi	LSTM-RNN-CTC	LSTM-RNN-CTC-Pretrained
Switchboard	%41.8 WER	-	-
TIMIT	-	%55.4 WER	%20.4 WER
NOAA	-	-	-
TIDigits	-	-	-

Table 2 - Results

6.4 Conclusion and Future Work

We have set up an RNN based on LSTM cells that use the CTC loss function. We trained two systems with medium to large size corpus and evaluated their performances based on Sclite scorer in terms of Word Error Rate. The results are compared with the HMM-based alternative. The experiment runs that we designed could only be completed partially due to limited time and resources, but we've reached a significant milestone for our future studies. For the next phases, we have produced source code and procedures that offer the preprocessing automation need. The source code is on our GitHub repository. There are significant opportunities for future studies such as data augmentation (e.g., volume, background noise, reverb, gap removal, overlay, resample of the audio files). Transfer learning also seems to be a game-changer in terms of efficiency in model training.

References

[Nassif] Nassif, A. B., Shahin, I., Attili, I., Azzeh, M., & Shaalan, K. (2019). Speech Recognition Using Deep Neural Networks: A Systematic Review. IEEE Access, 7, 19143–19165. https://doi.org/10.1109/ACCESS.2019.2896880

[Liu] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. Neurocomputing, 234(October 2016), 11–26. https://doi.org/10.1016/j.neucom.2016.12.038

[Koller] Koller, D., Friedman, N. (2009). Probabilistic Graphical Models: Principles and Techniques. United Kingdom: MIT Press.

[Graves] Graves, A., & Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. 31st International Conference on Machine Learning, ICML 2014, 5, 3771–3779.

[Li] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, Yifan Gong, and A. A. (2013). Recent Advances in Deep Learning for Speech Recognition at Microsoft. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 0–4. http://research.microsoft.com/apps/pubs/?id=188864

[Renals] Renals, S., Morgan, N., Bourlard, H., Cohen, M., & Franco, H. (1994). Connectionist Probability Estimators In Hmm Speech Recognition. IEEE Transactions on Speech and Audio Processing, 2(1), 161–174. https://doi.org/10.1109/89.260359

[Lamere] Lamere, P., Kwok, P., Walker, W., Gouvêa, E., Singh, R., Raj, B., & Wolf, P. (2003). Design of the CMU Sphinx-4 decoder. EUROSPEECH 2003 - 8th European Conference on Speech Communication and Technology, 1181–1184.

[ESPnet] Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Soplin, N. E. Y., Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., & Ochiai, T. (2018). ESPNet: End-to-end speech processing toolkit. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, 2018-September, 2207–2211. https://doi.org/10.21437/Interspeech.2018-1456

[DeepSpeech] Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., & Ng, A. Y. (2014). Deep Speech: Scaling up end-to-end speech recognition. 1–12. http://arxiv.org/abs/1412.5567 [Switchboard] Godfrey, John J., and Edward Holliman. Switchboard-1 Release 2 LDC97S62. Web Download. Philadelphia: Linguistic Data Consortium, 1993.

[TIDigits] R. Gary Leonard, and George Doddington. TIDIGITS LDC93S10. Web Download. Philadelphia: Linguistic Data Consortium, 1993.

[TIMIT] Garofolo, John S., et al. TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1. Web Download. Philadelphia: Linguistic Data Consortium, 1993.

[NOAA] Speech:NOAA @ foss.unh.edu. https://foss.unh.edu/projects/index.php/Speech:NOAA

[cs224s] Stanford CS224S / LINGUIST285 - Spoken Language Processing web.stanford.edu. http://web.stanford.edu/class/cs224s/

7 Annex: High Level Design

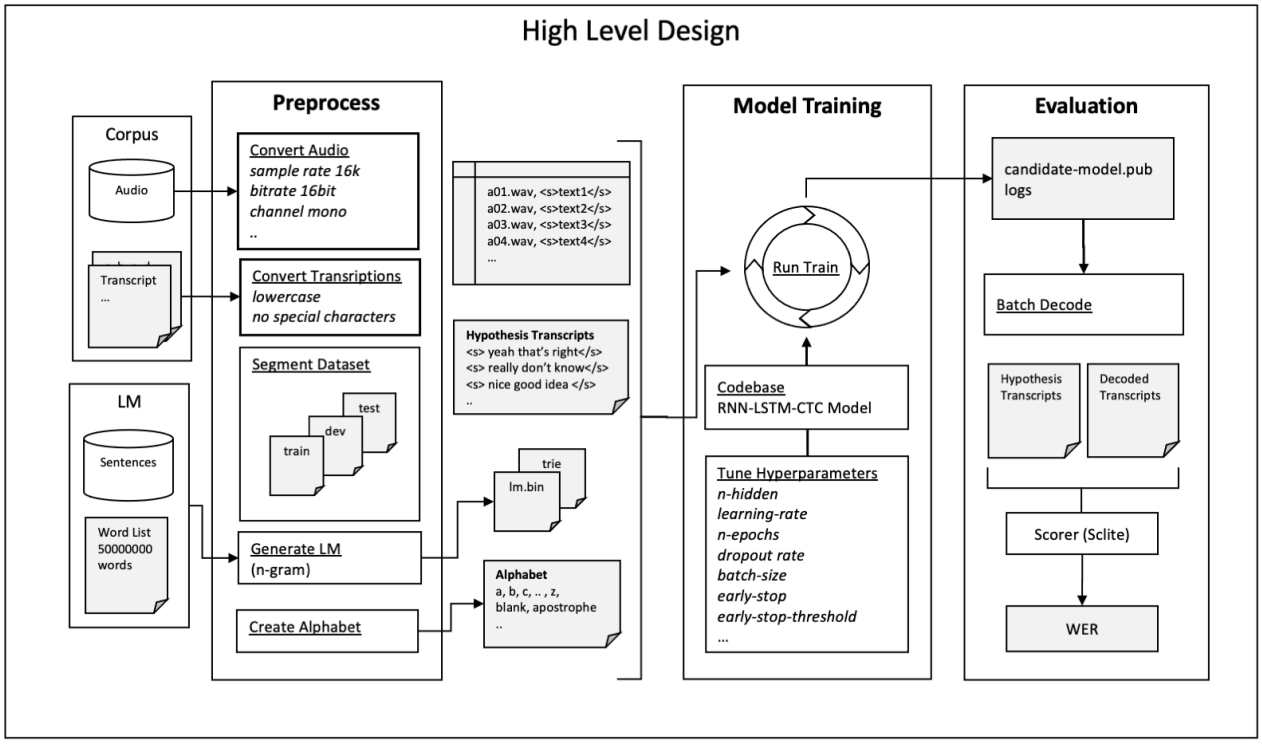

Figure6 - High Level Design

8 Annex: Codebase

The following repository offers preprocessing steps, model parameters, the data used during our experiments.
https://github.com/aygar/deeplearning