Speech Recognition with Deep Recurrent Neural Networks

Introduction:

The paper is an implementation paper of end to end Automatic Speech Recogition(ASR) system.
There are 2 algorithms which are explored with the paper.
i) Connectionist Temporal Classification (CTC)
ii)RNN Transducer

Let us look at the how ASR system has evolved over the years.

Till RNN was incorporated to Speech Recognition, HMMs were responsible to predict the phoneme/word from the input.
HMM is a statistical method where the probability of the output depends on the previous timestep. This can be applied for simpler problems[1].
RNN is a good replacement for HMM as it could store the context of the previous outputs better than HMM.
Due to the vanishing Gradient problem, RNNs are later replaced by LSTMs.
In order to predict the phoneme correctly, even the future context could be used effectively. So Bidirectional LSTM is used and the deep Bidirectional LSTMs are the state of the art architecture for Speech Recognition.

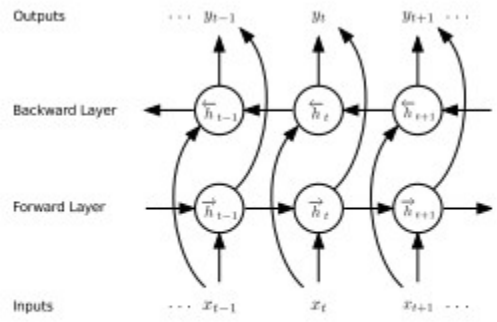
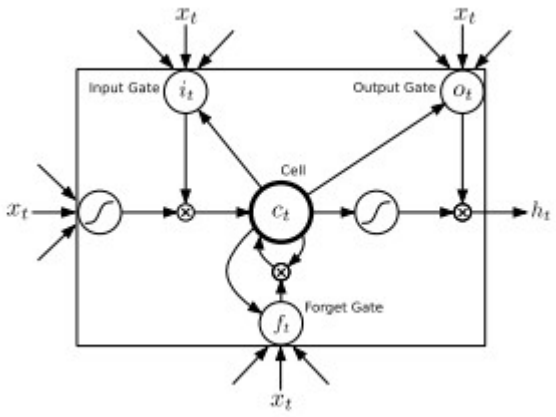


Fig. 2. Bidirectional RNN

Each LSTM has a memory cell implemantation. So the memory is regulated throughout the network wherever it is most necessary by the network itself. The LSTM works as follows:

$$
\begin{aligned}
i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \\
c_t &= f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \\
o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$



Bidirectional RNN:

As bidirectional RNN has seperate LSTM cells for backward and Forward directions, we have to merge the ouput of both directions into single output at every timestep as shown below:

$$\overrightarrow{h}_t = \mathcal{H}\left(W_{x\overrightarrow{h}}x_t + W_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right)$$
$$\overleftarrow{h}_t = \mathcal{H}\left(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t+1} + b_{\overleftarrow{h}}\right)$$
$$y_t = W_{\overrightarrow{h}y}\overrightarrow{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y$$

Preprocessing:

There are many feature extraction methods used on input speech signal:

Perceptual Linear Prediction (PLP)
Relative spectra filtering of log domain coefficients PLP (RASTA-PLP)
Linear predictive coding (LPC)
Predictive cepstral coefficients (LPCC)
Mel scale cepstral analysis (MEL)
Mel-frequency cepstral coefficients (MFCC)
Outdated Power spectral analysis (FFT)

First order derivative (DELTA)
Energy normalization etc

where MFCC is the most popular even today.
We can even use the raw signals, but is not yet able to produce state of the art results.

Dataset:

The dataset which is used in the paper is called the TIMIT dataset.
TIMIT contains broadband
recordings of 630 speakers of 8 major dialects of American English,
each reading 10 phonetically rich sentences. The TIMIT corpus includes
time-aligned orthographic, phonetic and word transcriptions as well as
a 16-bit, 16kHz speech waveform file for each utterance. Corpus design
was a joint effort among the Massachusetts Institute of Technology
(MIT), SRI International (SRI) and Texas Instruments, Inc. (TI)

Let us explore the end to end approaches in the paper.

1.Connectionist Temporal Classification:

Training :

Consider we have a RNN network of 10 timesteps.

Speech Recognition is the task where we may not have output at every timestep. So the annotated output will have only the phonemes which are uttered. So traditional algorithm doesn't work.

So CTC is an algorithm which is sensitive to varied mapping between input and target.
CTC explores all the possible ways we can align the target to all timesteps.

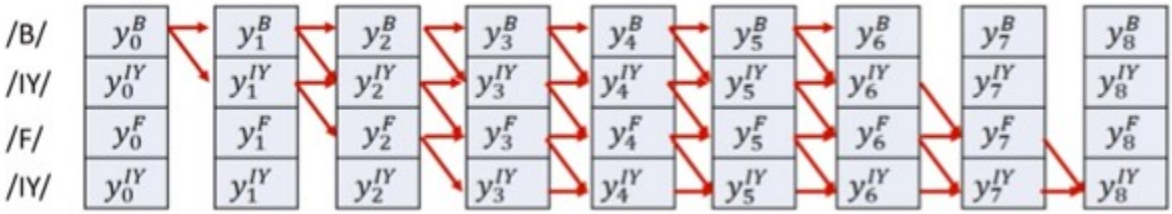
Ex: If the target ouput is '/he/' '/el'/ '/oh/' making it the word "hello"

If we have to allign this target output to every timestep, there are multiple ways in which we can align.
Ex: '-' '/he/' ''/he/' '/he/' '-' '/el/' '/el/' '/el/' '-' '/oh/' OR
'/he/' '-' '-' '/el/' '/el/' '/el/' '-' '/oh/' '/oh/' '/oh/' OR other possible paths

To get all the paths , dynamic programming algorithm called Viterbi algorithm is used where
We first initialize the weights of the network.
We run the forward prop.
At the end of forward prop we get the probabilities of all the phonemes at each timestep.

From this probability vectors we only extract out the probabilities of unique phonemes in our target sequence

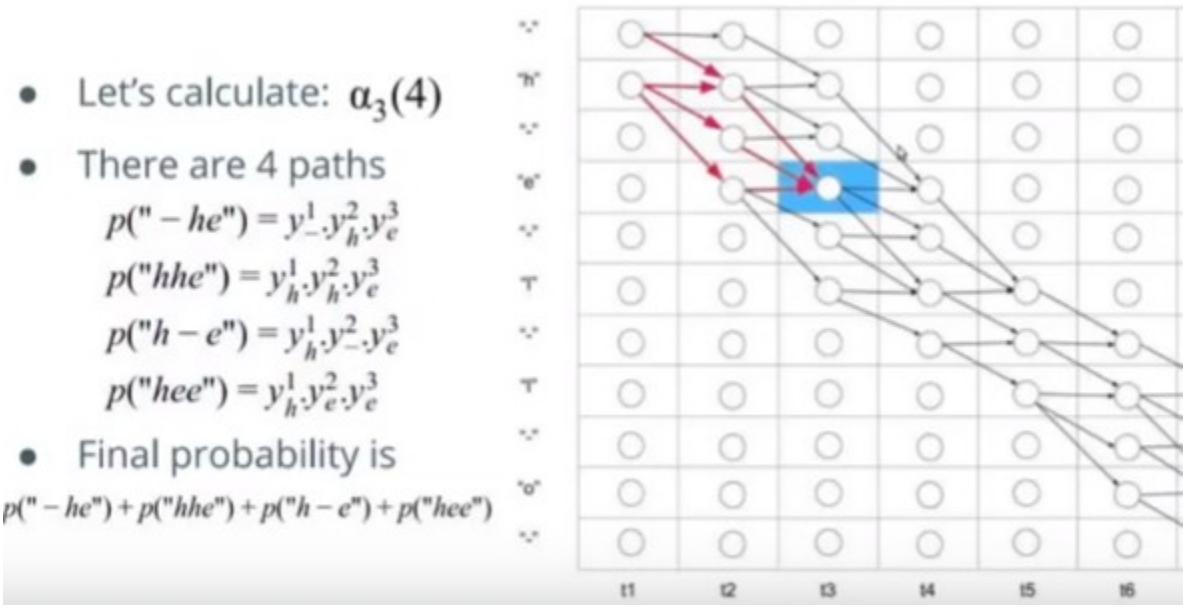
And then we constrain the extracted probability to some rules which could generate all possible allignment paths for our target.



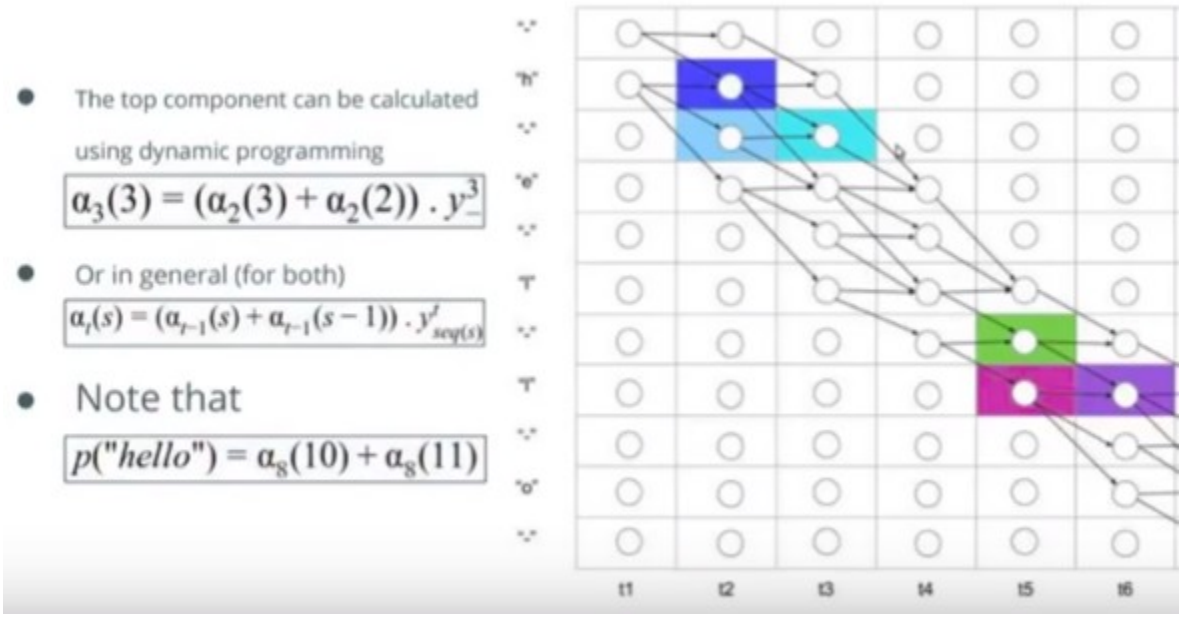
Now At each step we calculate loss considering each symbol in the valid path at every timestep

Ex: At 1st timestep only '/B/' can occur . So we calculate the probability of '/B/' occuring for a valid path.
This is calculated as α and β where alpha gives the probability of all the symbols before the phoneme '/B/' for all valid paths and beta gives the probabililty of path that can occur after symbol '/B/'

for a different example, if we have to calculate α for 3rd timestep for the symbol 'e' then it is calculated as sum of probability of all paths that pass through our cuurent symbol



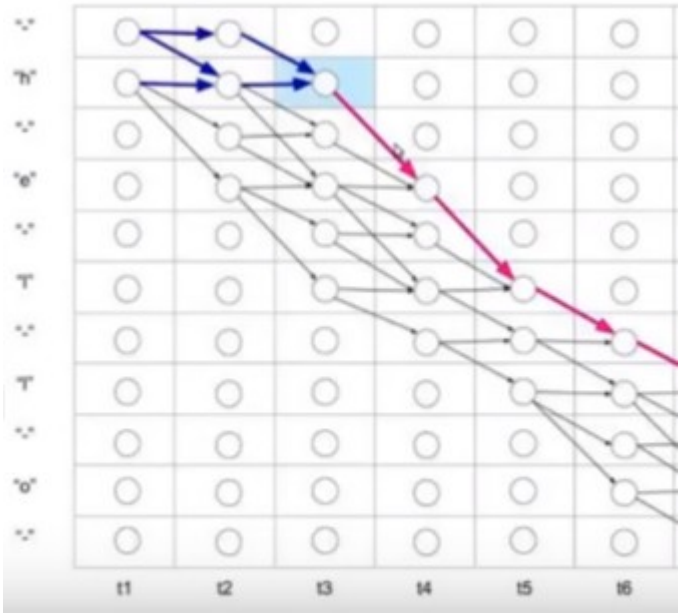
We can also apply dynamic programming here to calculate this quickly by calculating α of previous timesteps as shown below.



Beta is calculate as the same but for the symbols which occur after our current target symbol for all valid paths.

We can say total probability of symbol that can occur at time t is
(alpha * beta) /probability of y symbol

Ex: Consider the below graph.



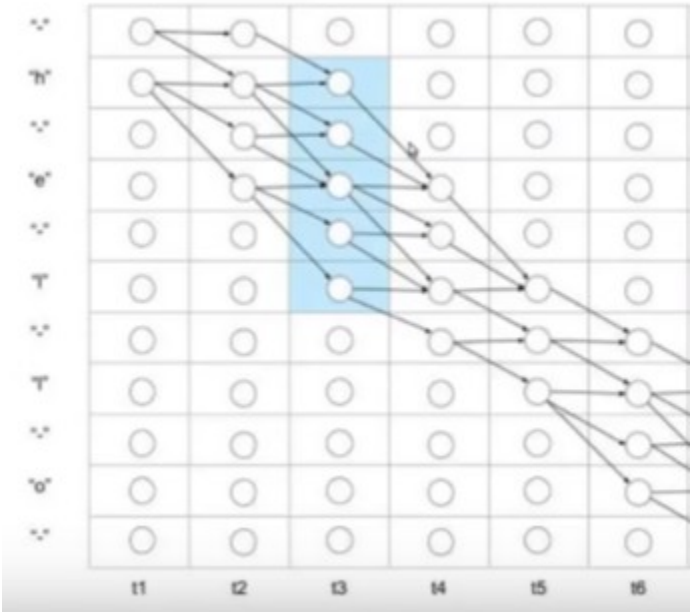
If we have to calculate the probability of h at time step 3 i.e. the blue box in the pic,
the formulae will be :

$$\frac{\alpha_3(2).\beta_3(2)}{y_h^3}$$

Now after calculating probabilities of all the symbols at every timestep which belongs to valid path,
we add all those probabilities as P and then take -log(P) as loss.

We calculate derivative for this and update weights at every of valid symbols.

Ex: for the 3rd timestep, if we have to calculate loss, we add the total probabilities of all these blue boxes and that becomes P



$$p(\text{"hello"}) = \sum_{s=2}^{6} \frac{\alpha_3(s).\beta_3(s)}{y^3_{seq(s)}}$$

Now we can apply Divergence by the formula:

Div = - log (p(“hello”))

The Backprop algorithm can be applied to this as below:

$$\frac{\partial(-ln\, p(\text{"hello"}))}{\partial y^t_k} = \frac{-1}{p(\text{"hello"})} \frac{\partial p(\text{"hello"})}{\partial y^t_k}$$

where

$$\frac{\partial p(\text{"hello"})}{\partial y^t_k} = \frac{-1}{{y^t_k}^2} \sum_{s:seq(s)=k} \alpha_t(s).\beta_t(s)$$

Inference:

At the inference time, we can decode the probability vectors using either Greedy Search Decoding or Beam Search Decoding or any Grammer based decoding.

2.RNN Transducer:

The idea behind RNN transducer is that CTC network only give output probability based on input.
It doesn't consider to output some phoneme based on the previous phoneme.
So this makes CTC network less robust because it could output non meaningful sequences.
Therefore it requires a language model to correct these errors.

As we are trying to avoid most of the external components here, So we define 2 seperate models.

First is called Transcription Network which is a CTC network

Then the second is called a Prediction Network

Prediction Network:

The prediction network takes y as input and gives out modified y such that now the phoneme sequence will be more meaningful because at each timestep, the output phoneme depends on the first phoneme.
So this somewhat correspond to the language model.

An extension to this is proposed in this paper where the hidden layer ouputs of both Transcription network and Prediction network is used to predict the final output as shown below:

$$
\begin{aligned}
l_t &= W_{\overrightarrow{h}^N l}\overrightarrow{h}_t^N + W_{\overleftarrow{h}^N l}\overleftarrow{h}_t^N + b_l \\
h_{t,u} &= \tanh\left(W_{lh}l_{t,u} + W_{pb}p_u + b_h\right) \\
y_{t,u} &= W_{hy}h_{t,u} + b_y
\end{aligned}
$$

where 'l' is the hidden layer output of Transcription network and 'p' is the hidden layer output of Prediction network.

Two regularisers were used in this paper: early stopping and weight noise

These algorithms prove to be the state of the art for end to end ASR system at 2013 as shown by the Experimental Results in the paper

References:

Hidden Markov Models: https://www.youtube.com/watch?v=kNloj1Qtf0Y&t=3s

Sequence Modelling with CTC: https://distill.pub/2017/ctc/

CTC : TDLS https://www.youtube.com/watch?v=UMxvZ9qHwJs

RNN Transducer: A. Graves, “Sequence transduction with recurrent neural networks,” in ICML Representation Learning Workshop, 2012.