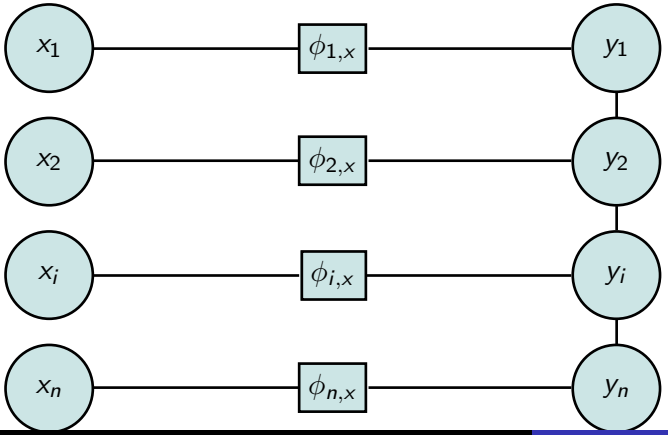
Lecture 25a: Lego World of Deep Learning, Recurrent Neural Networks, LSTMs, CTC, etc

Instructor: Prof. Ganesh Ramakrishnan

Recap: Linear Conditional Random Fields (CRF)

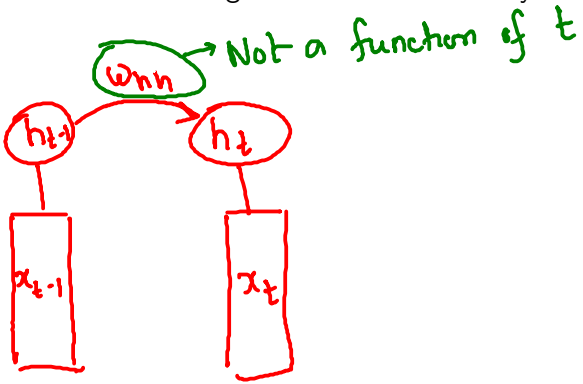
Just as CRF was an extension of Logistic Regression (LR) can Neural Networks (cascade of LRs) be extended to sequential output?

inputs $x-$potentials classes & $y-$potentials $\phi_{i,y}$



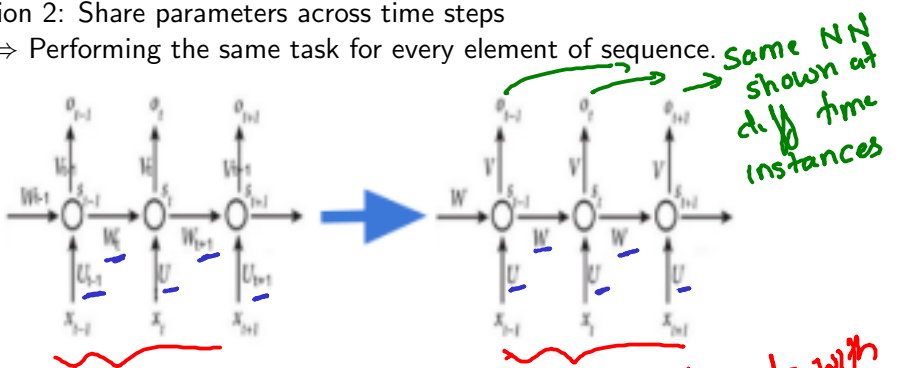
Recurrent Neural Network (RNN) Intuition

- Recall: In CNN we used the trick of common parameters for many neurons
- RNN intuition 1: We want a neuron's output at time t to depend on its state s at time $t-1$
- RNN intuition 2: Share parameters across time steps
- Recurrent $\Rightarrow$ Performing the same task for every element of sequence.



Recurrent Neural Network (RNN) Intuition

- Recall: In CNN we used the trick of common parameters for many neurons
- RNN intuition 1: We want a neuron's output at time t to depend on its state s at time $t-1$
- RNN intuition 2: Share parameters across time steps
- Recurrent $\Rightarrow$ Performing the same task for every element of sequence.

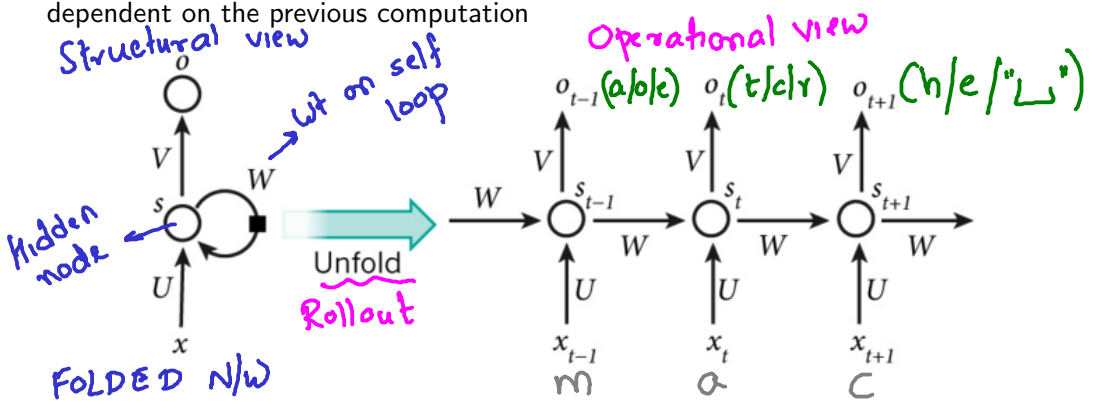


Tutorial 8: Problem 7

Try the text generation RNN (Recurrent Neural Network) demo at `http://www.cs.toronto.edu/~ilya/rnn.html`. State any interesting observations. How would you improve the performance of the RNN?

RNN: Compact representation

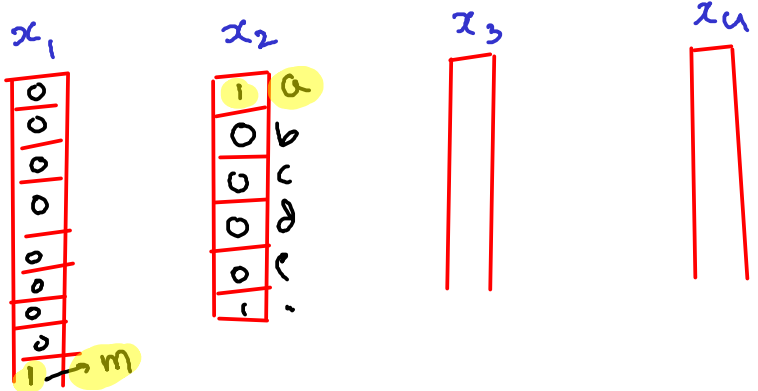
- Generalization of Neural networks to Sequential tasks such as *language modeling*, *word prediction*, *etc.*.
- Perform the same *task* for every element of the sequence, with the output being dependent on the previous computation



A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature

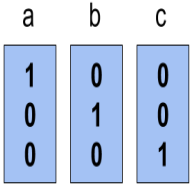
RNN: One Hot Encoding for Language Model

- With 3 characters in vocabulary, a,b and c, what would be the best encoding to inform each character occurrence to the network?
- One Hot Encoding: Give a unique key k to each character in alpha-numeric order, and encode each character with a vector of vocabulary size, with a 1 for the k^{th} element, and 0 for all other elements.

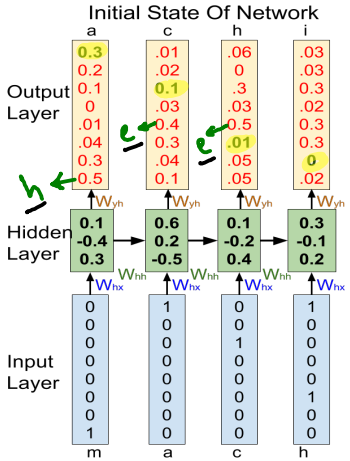


RNN: One Hot Encoding for Language Model

- With 3 characters in vocabulary, a,b and c, what would be the best encoding to inform each character occurrence to the network?
- One Hot Encoding: Give a unique key k to each character in alpha-numeric order, and encode each character with a vector of vocabulary size, with a 1 for the k^{th} element, and 0 for all other elements.



RNN: Language Model Example with one hidden layer of 3 neurons



Model gets a chance to correct its wts through back propagation through time (BPTT)

⇒ Correcting the green characters to the highted modes

RNN: Language Model Example with one hidden layer of 3 neurons

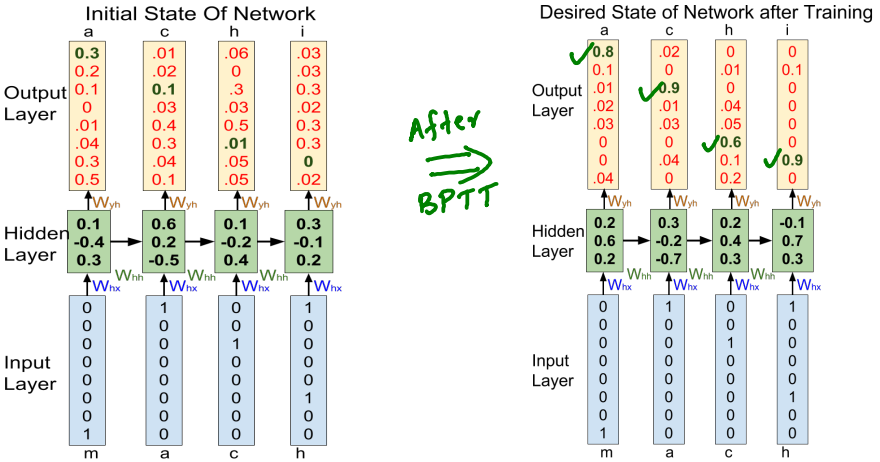


Figure 1: Unfolded RNN for 4 time units

RNN: Equations (Used for speech to text by Youtube)

- $h_t = g(W_{hh}h_{t-1} + W_{hx}x_t + b_h)$
  - Activation function g could be sigmoid σ or its extension to multiclass called softmax[1], tanh $\left(\frac{1-e^{-2x}}{1+e^{-2x}}\right)$ which is simply a scaled[2] and shifted version of the sigmoid function
  - A network may have combination of different activation functions[3]
- $y_t = W_{yh}\ h_t$
- The new (present) hidden state depends upon the previous hidden state(s) and the present input.
- The present output depends upon present hidden state (and in turn upon previous hidden states).

---

[1] Tutorial 7

[2] $tanh(x) = 2\sigma(2x) - 1$

[3] `http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano`

Back Propagation Through Time: BPTT Illustration

- $h_1 = g(W_{hh}h_0 + W_{hx}x_0 + b_h)$, initialize h_0 and x_0 as zero vectors.
- At $t = 2$ we present x_2 as 'a' at input and desire y_2 as 'c' at output in one hot encoded form as shown previously
- $h_2 = g(W_{hh}h_1 + W_{hx}x_1 + b_h)$
- At $t = 3$, $x_3 =$ 'c', y_3 we desire is 'h'.
- $y_3 = W_{yh}\ g(W_{hh}h_2 + W_{hx}x_2 + b_h)$

Back Propagation Through Time: BPTT Illustration

Forward prop.

- $h_1 = g(W_{hh}h_0 + W_{hx}x_0 + b_h)$, initialize h_0 and x_0 as zero vectors.
- At $t = 2$ we present x_2 as 'a' at input and desire y_2 as 'c' at output in one hot encoded form as shown previously
- $h_2 = g(W_{hh}h_1 + W_{hx}x_1 + b_h)$
- At $t = 3$, x_3 = 'c', y_3 we desire is 'h'.
- $y_3 = W_{yh}\ g(W_{hh}h_2 + W_{hx}x_2 + b_h)$
- Put h_1 and h_2 in the last equation and then perform gradient descent on weights (*i.e.*, **tune weights** through back propagation) to fit y_3 first corresponding to vectors x_3, x_2 and x_1.
- Similarly use h_1 in equation for y_2 and **tune weights** to fit y_2 corresponding to vectors x_2 and x_1.
- Then **tune weights** for y_1.

Backprop

$y_3 \rightarrow h_2 \rightarrow y_2 \rightarrow$ - - -

RNN Parameters

- In previous example, we used
  1. **sequence length** of 4, (that is, the number of time steps to unroll the RNN)
  2. **batch size** of 1, (that is, the number of input vectors presented at a single time step to the RNN)
- We can vary these parameters according to the task at hand.
- One hot encoding is the best suited encoding for such tasks while training the neural networks.

RNN Limitations

BPTT involves lot more multiplication than normal backprop.

- **Vanishing gradient:** When training networks with long range dependencies, the influence of the given input decays exponentially as it cycles around the network's recurrent connections.
- Gradient vanishes especially when we use sigmoid function and several gradient values v with $|v| < 1$, get multiplied during BPTT to give a zero.
- Instead, if we used an alternative function that gives value > 1 as output, we will face the problem of **'exploding gradient'**.

RNN: Vanishing Gradient Problem

The sensitivity(derivative) of network w.r.t input($@t = 1$) decays exponentially with time, as shown in the unfolded (for 7 time steps) RNN below. Darker the shade, higher is the sensitivity w.r.t to x_1.

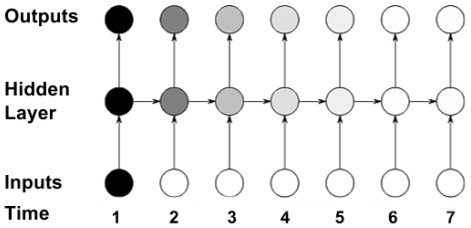


Image reference: Alex Graves 2012.

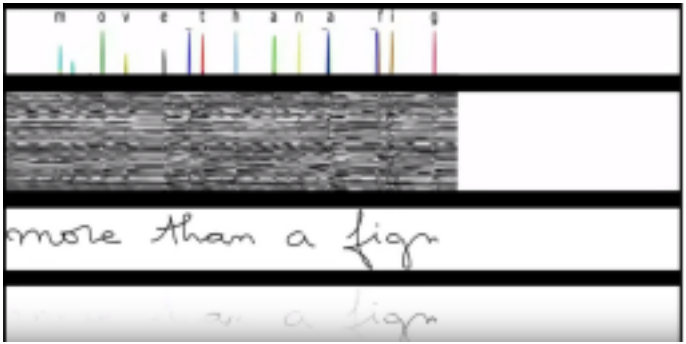
Long Short-Term Memory (LSTM) Intuition

- Learn when to propagate gradients and when not, depending upon the sequences.
- Use the memory cells to store information and reveal it whenever needed.
- I live in **India**.... I visit **Mumbai** regularly.
- For example: Remember the context "India", as it is generally related to many other things like language, region etc. and forget it when the words like "Hindi", "Mumbai" or End of Line/Paragraph appear or get predicted.

Recap: The Lego Blocks in Modern Deep Learning

1. Depth/Feature Map
2. Patches/Kernels (provide for spatial interpolations) - **Filter**
3. Strides (enable downsampling)
4. Padding (shrinking across layers)
5. Pooling (More downsampling) - **Filter**
6. **RNN and LSTM** (Backpropagation through time and Memory cell)
7. **Connectionist Temporal Classification**
8. Embeddings (Later, with unsupervised learning)

Demonstration of Alex Graves's system working on pen coordinates



1. Top: Characters as recognized, without output delayed but never revised.
2. Second: States in a subset of the memory cells, that get reset when character recognized.
3. Third: Actual writing (input is x and y coordinates of pen-tip and up/down location).
4. Fourth: Gradient backpropagated all the way to the xy locations. Notice which

LSTM Equations

- $f_t = \sigma(W_{xf} x_t + W_{hf} h_{t-1} + W_{cf} c_{t-1} + b_f)$
- $i_t = \sigma(W_{xi} x_t + W_{hi} h_{t-1} + W_{ci} c_{t-1} + b_i)$
- We learn the forgetting (f_t) of previous cell state and insertion (i_t) of present input depending on the present input, previous cell state(s) and hidden state(s).

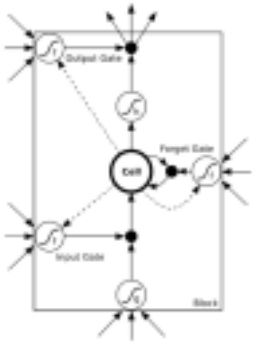


Image reference: Alex Graves 2012

LSTM Equations

- $c_t = f_t c_{t-1} + i_t tanh(W_{hc} h_{t-1} + W_{xc} x_t + b_c)$
- The new cell state c_t is decided according to the firing of f_t and i_t.

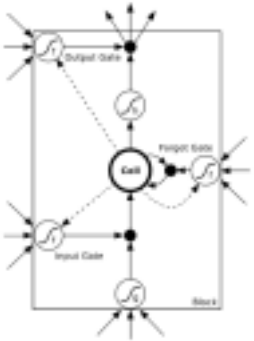


Image reference: Alex Graves 2012.

LSTM Equations

- $c_t = f_t c_{t-1} + i_t tanh(W_{hc} h_{t-1} + W_{xc} x_t + b_c)$
- $f_t = \sigma(W_{xf} x_t + W_{hf} h_{t-1} + W_{cf} c_{t-1} + b_f)$
- $i_t = \sigma(W_{xi} x_t + W_{hi} h_{t-1} + W_{ci} c_{t-1} + b_i)$
- Each gate is a vector of cells; keep the constraint of W_{c*} being diagonal so that each element of LSTM unit acts independently.
- $o_t = \sigma(W_{xo} x_t + W_{ho} h_{t-1} + W_{co} c_{t-1} + b_f)$
- $h_t = o_t \ tanh(c_t)$

LSTM Gradient Information remain preserved

The opening 'O' or closing '-' of input, forget and output gates are shown below, to the left and above the hidden layer respectively.

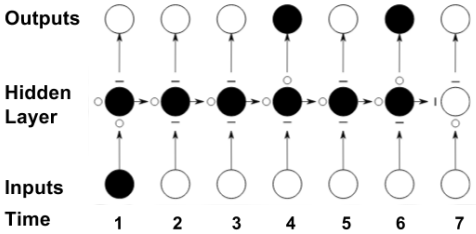


Image reference: Alex Graves 2012.

LSTM V/S RNN results on Novel writing

A RNN and a LSTM, when trained appropriately with a Shakespeare Novel write the following output (for few time steps) upon random initialization.

RNN's Novel	LSTM's Novel
kentages, whom falled me, Compassion, that the wind of England topsued To grief all go. AUFIDIUS: Marry, it were any graven, And high agains. PAULINA: Natus my plateniss, this are the sitty, but continue stillen sonsable Tybalt. GREMIO: God sues we charpet. Since this is shoulders to the king. KING EDWARD IV: Look, myself. Sir Juige: ...	Monsum and Parustage, is no neves Desirand is a death barshers: Her much. LEONTES: JULIET: O Wanswhat that she were a pituhpy, that nothing. Second Lord: They are of my foul me: power and your aclibing. Lore and the glatishal indeed, stamp: hell, if procing dages. LADY GREY: No noter them beligsew the burges That Enrilith these out with her good ...

Sequence Labeling

- The task of labeling the sequence with discrete labels. Examples: Speech recognition, handwriting recognition, part of speech tagging.
- Humans while reading/hearing make use of context much more than individual components. For example:- Yoa can undenstard dis, tough itz an eroneous text.
- The sound or image of individual characters may appear similar and may cause confusion to the network, if the proper context is unknown. For example: "in" and "m" may look similar whereas "dis" and "this" may sound similar.

Type of Sequence Labeling Tasks

- Sequence Classification: Label sequence is constrained to be of unit length.

Type of Sequence Labeling Tasks

- Segment Classification: Target sequence consist of multiple labels and the segment locations of the input is known in advance, e.g. the timing where each character ends and another character starts is known in a speech signal.

- We generally do not have such data available, and segmenting such data is both tiresome and erroneous.

Type of Sequence Labeling Tasks

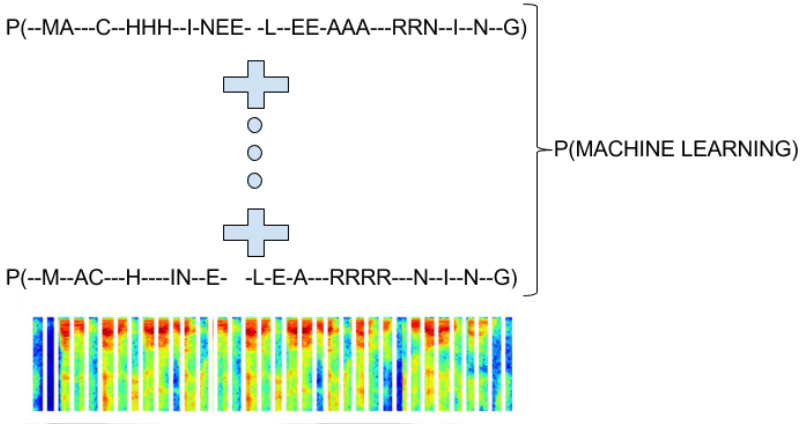
- Temporal Classification: Tasks in which temporal location of each label in the input image/signal does not matter.

- Very useful, as generally we have higher level labeling available for training, e.g. word images and the corresponding strings, or it is much easier to automate the process of segmenting the word images from a line, than to segment the character images from a word.

Connectionist Temporal Classification (CTC) Layer

- For temporal classification task: length of label sequence ¡ length of input sequence.
- CTC label predictions at any time in input sequence.
- Predict an output at every time instance, and then decode the output using probabilities we get at output layer in vector form.
- e.g. If we get output as sequence "–m–aa-ccch-i-nee– -lle–a-rr-n-iinnn-g", we will decode it to "machine learning".
- While training we may encode "machine learning" to"-m-a-c-h-i-n-e--l-e-a-r-n-i-n-g-" via C.T.C. Layer.

CTC Intuition [Optional]



CTC Intuition [Optional]

NN function : f(x_T) = y_T

- x_T: input image/signal x of length T.
  - For image: each element of x_T is a column(or its feature) of the image.
- y_T: output sequence y of length T.
  - each element of y_T is a vector of length —A'—(where A' = A∪"-" i.e. alphabet set ∪ blank label).

ℓ_U : Label of length U(¡T).
Intuition behind CTC:

- generate a PDF at every time-step t ∈ 1,2,...,T.
- Train NN with objective function that forces Max. Likelihood to decode x_T to ℓ_U(desired label).

CTC Layer: PDF [Optional]

$P(\pi—x) = \prod_{t=1}^{T} y_t(\pi_t)$

- path π : a possible string sequence of length T, that we expect to lead to ℓ. For example: -p-a-t-h-", if ℓ = "path".
- y_i(n): probability assigned by NN when character n($\in$ A') is seen at time i. "-" is symbol for blank label.
- π_t : t^{th} element of path π.

$P(\ell—x) = \sum_{label(\pi)=\ell} P(\pi—x)$

CTC Layer: PDF [Optional]

$P(\ell|x) = \sum_{label(\pi)=\ell} P(\pi|x) = \sum_{label(\pi)=\ell} \prod_{t=1}^{T} y_t(\pi_t)$

- Question: What could be possible paths of length $T = 9$ that lead to $\ell =$ "path"?
- Answer:
- Question: How do we take care of cases like $\ell =$ "Mongoose"?
- Answer:

CTC Layer: PDF [Optional]

$P(\ell|x) = \sum_{label(\pi)=\ell} P(\pi|x) = \sum_{label(\pi)=\ell} \prod_{t=1}^{T} y_t(\pi_t)$

- Question: What could be possible paths of length T = 9 that lead to ℓ = "path"?
- Answer: -p-a-t-h-", pp-a-t-h-", -paa-t-h-", -ppa-t-h-", -p-aat-h-" etc.
- Question: How do we take care of cases like ℓ = "Mongoose"?
- Answer: We change ℓ = "Mongoose" to ℓ = "Mongo-ose".
- Question: During training ℓ is known, what to do at testing stage?
- Answer: Next Slide.

CTC Layer: Forward Pass Decoding [Optional]

$P(\ell|x) = \sum_{label(\pi)=\ell} P(\pi|x) = \sum_{label(\pi)=\ell} \prod_{t=1}^{T} y_t(\pi_t)$

Question: During training ℓ is known, what to do at testing stage?

1. Brute force: try all possible ℓs, all possible πs for each ℓ to get $P(\ell|x)$ and choose best ℓ.

   - Rejected as expensive.

2. Best Path Decoding - most likely path corresponds to the most likely label.
   - $P(A1) = 0.1$, where A1 is the only path corresponding to label A.
   - $P(B1) = P(B2) = \ = P(B10) = 0.05$, where B1..B10 are the 10 paths corresponding to label B.
   - Clearly B is preferable over A as $P(B|x) = 0.5$.
   - But Best Path Decoding will select A.

   - Rejected as inaccurate.

3. Prefix Search Decoding - NEXT

CTC Layer: Prefix Search Decoding [Optional]

1. Initialize prefix p* = ϕ; //p* is prefix of /*ell*
2. P(ϕ) = 1; // as ϕ is prefix of every word.
3. try p_{new} = p* + k, for all k $\in$ A$\cup${eos}; // + represent concatenation.
4. Maintain L_p: list of growing prefixes; // —A— + 1 new values per iteration.
5. Maintain P_p: list of probabilities of corresponding elements in Lp; //How to find P(p*+ k)? Next Slide.
6. if P(p*+eos) ¿= max(P_p): stop and go to step 8;
7. else: update p* with prefix having max prob. and repeat from step 3;
8. p* is the required prefix.

In practice beam-search is used to limit the exponentially growing Lp and make the decoding faster.

CTC Layer: Prefix Search Decoding

Consider the DAG shown below with A = {x,y} and e representing the end of string. The steps a-f represent the path followed by Prefix Search Decoding Algorithm.

- What ℓ would the Best Path Decoding Produce?
- What ℓ would the Prefix Search Decoding Produce?