CTC

i) Introduce ∈ blank token, can be removed

h h e ∈ ∈ l l ∈ l ∈ o o ∈ → valid

⇓

h e l l o

(h ∈ h) e ∈ ∈ l ∈ l (l ∈ l) ∈ o o o → invalid → bad alignment

e l l ∈ o → invalid missing he

Properties

→ Monotonic order (left to right) :- (no reordering)

→ Many to one

→ len(Y) ≤ len(X)

Problem :-

→ A sequence $X_1, X_2 \dots X_n$ goes in

→ Different sequence $Y_1, Y_2 \dots Y_M$ comes out

Generally $N \neq M$

→ Speech goes in → words come out

English → german

I ate an apple → Ich habe einen apfel gegessen

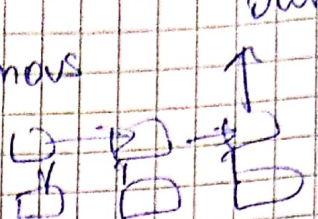
extra

Order changed & extra word added

Case1:- With alignment

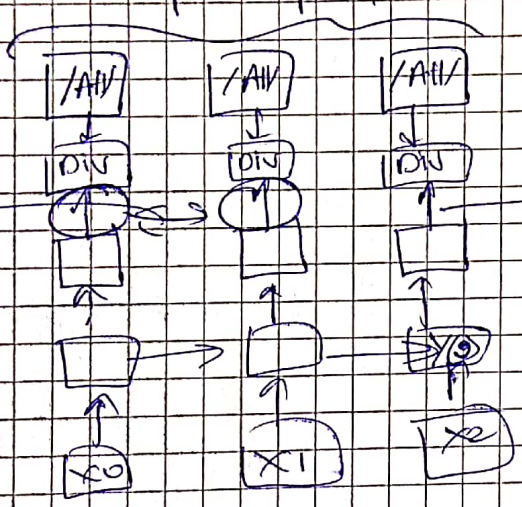
→ Happens in same order but are asynchronous

Ex:- speech recognition



→ Ignores output at intermediate steps (Ex:- at time 0, 1 .. 2 etc)

→ Why can't we use same label through out each timestep even if we don't have complete input?



**

→ IT won't work in Question - answer scenario

What is the color of (sky) → here until sky comes in color can be of any thing only after we see sky we should answer blue else we would get blue for other objects

What is the color of (tree)? → blue as during training we pictured same answer all over

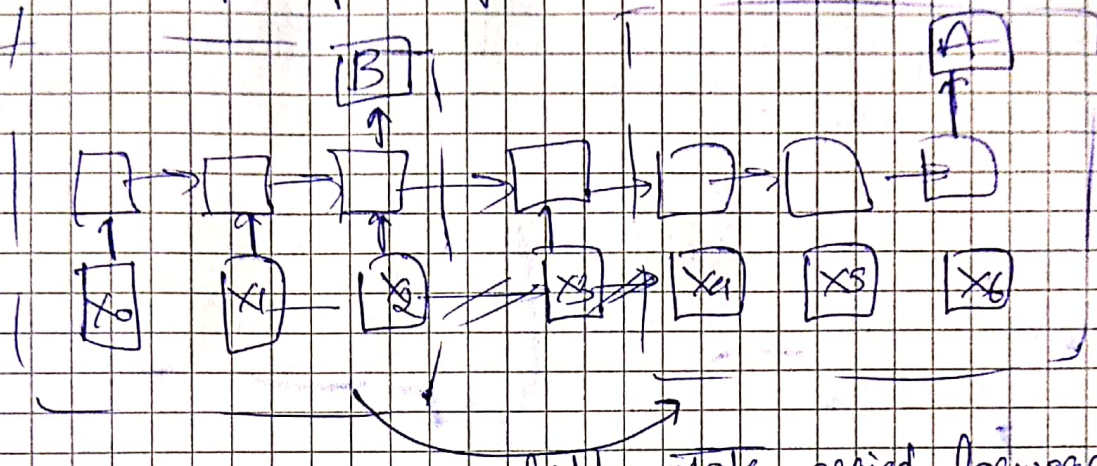
**

→ Naive Assumption :- Provided we have enough training data above method works

More complex problem

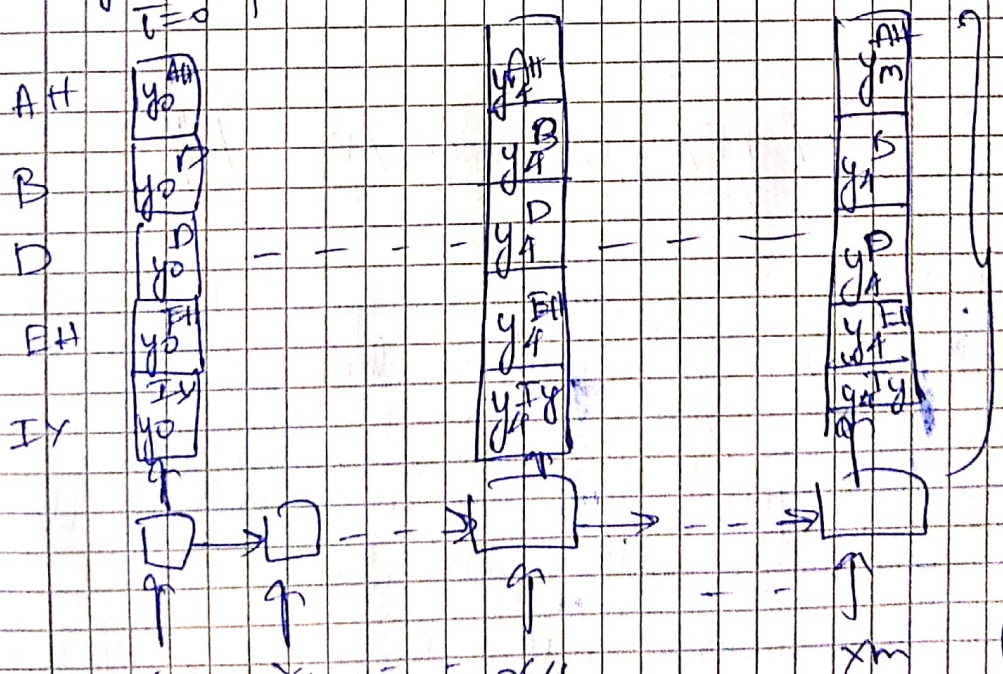
given a sequence of input asynchronously output sequence of symbols

This is just a concatenation of many copies of simple output attend of input sequence



** → We also don't know when to output symbols during forward prob. Hence make n/w o/p at each time step even if it is blank. It is not clear when each phoneme ends as every phoneme has some prob at each time step

$y_t^D = \text{prob}(s_t = D \mid X_0 \ldots X_t)$ $t=0$



** Don't know 'k'

Find most likely symbol sequence given input

$s_0 \ldots s_{k-1} = \underset{s_0' \ldots s_{k-1}'}{\text{argmax}}\ \text{prob}(s_0' \ldots s_{k-1}' \mid X_0 \ldots X_{n-1})$

** Option 1:
What happens if i pick only letter with most prob at each time step?
→ We would end up having output sequence with same length as input (Normally len(op) ≤ len(ip))
→ Results in cyclical & less diversity o/p
→ Example: Cannot represent or distinguish b/w repeated characters v/s extended characters Collapses all

** Option 2: Impose external constraints like op should be dictionary & it should have some min length. We refer this process as decoding

How To learn?

1) Within some span of letter, we can assume same symbol repeats

$$Div = -\sum_{I} \log Y(t_i, \text{symbol}_I)$$

Option 1

/g/ /g/ /F/ /F/ /F/ /F/ /IY/ /IY/ /D/

Actual input : g g F F I Y D
Produced output : g F I Y D
} Cannot distinguish b/w extended & repeated chars

→ But we can use maximum likelyhood and somehow increase decrease the chance of getting correct output with repeated chars
→ Generated sequence gFIYD is invalid as there is no word in existence check in dictionary

/a/ /a/ → a1 a2 a3 → a1 a2 a3
split phoneme repeat uniqueness maintained

1/6/20

CTC continued (Bhiksha Raj)

gradient wrt to th o/p vector Y_t

$$\nabla_{Y_t} Div = \left[0 \quad 0 \quad \cdots \quad \frac{-1}{Y(t, symbol_i)} \quad 0 \quad \cdots \quad 0\right]$$

zeros except at component corresponding to target

what if no timing is provided?

Soln: Guess the alignment

Initialize: Randomly or based on some heuristic or any other

Iterate: Train n/w over current alignment

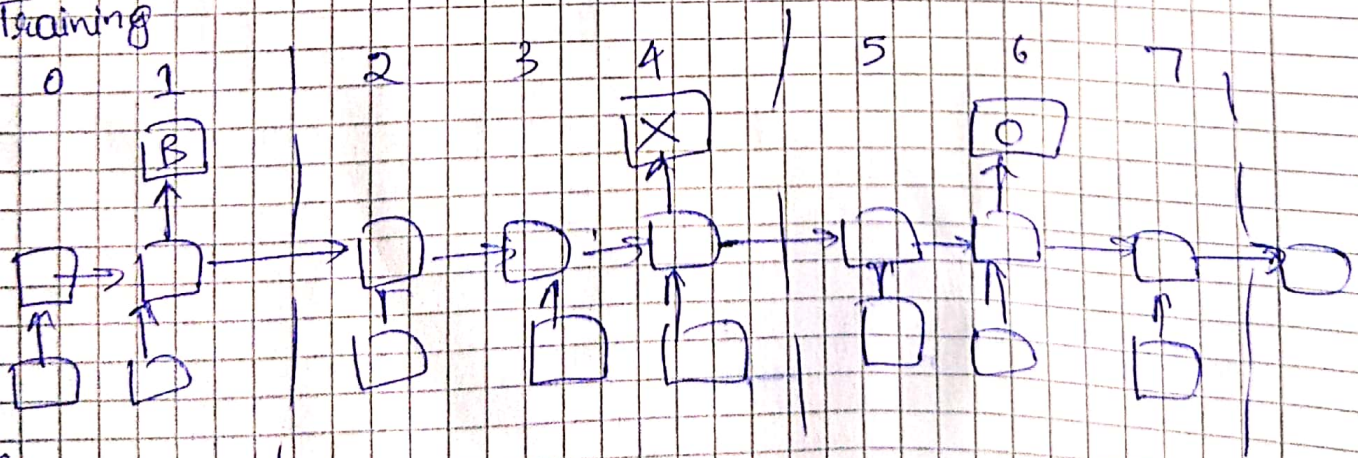
- Reestimate alignment for each training instance by using decoding method

---

Decoding: Output from RNN obtained, This can be addressed with option 1, 2

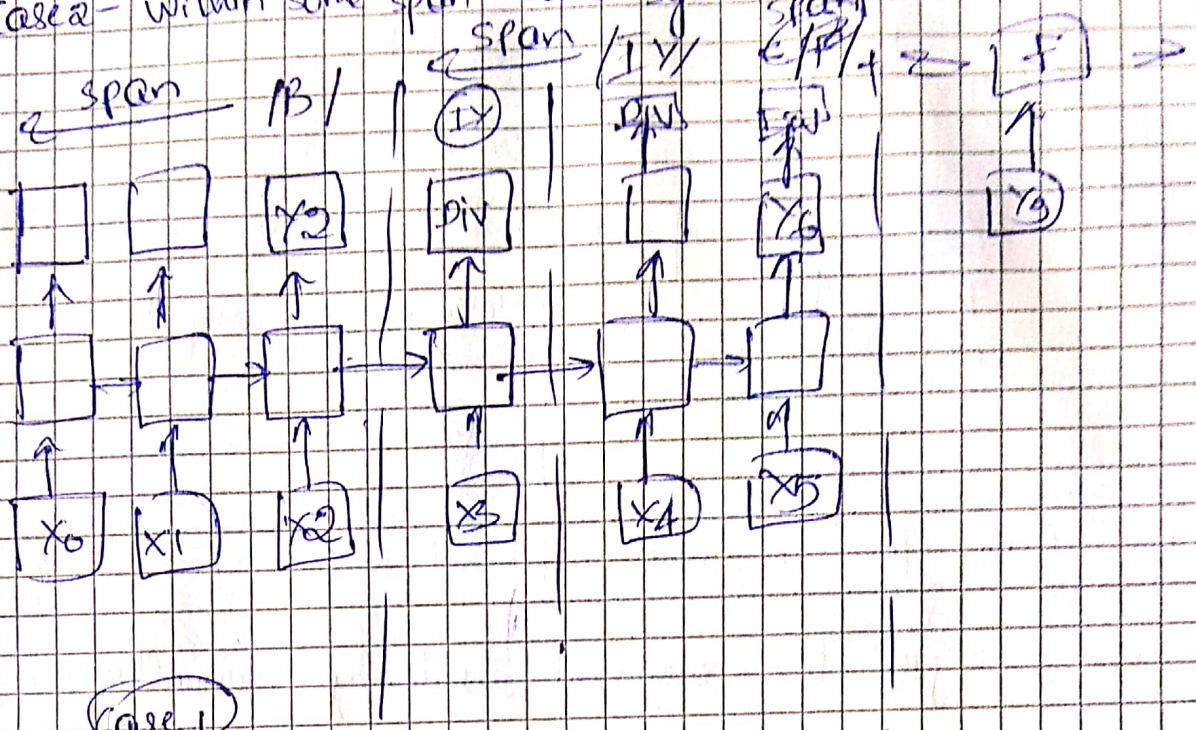
Bigger issue is training

Training



If I know at which timestep symbol occurs (1, 4, 6) we can compute crossentropy, gradients only at those locations

Case 2 - Within some span same symbol repeats



Case 1

Divergence = Xent (Y2, B) + Xent (Y4, IY) + Xent (Y6, F) + Xent (Y8, IY)

at individual 1 step

or repeat symbols during their span

Case 2 V = Xent (Y1, B) + Xent (Y2, B) + Xent (Y3, IY) + Xent (Y4, IY) + Xent (Y5, IY) + Xent (Y6, F) + Xent (Y7, F) + ... + Xent (Y9, IY)

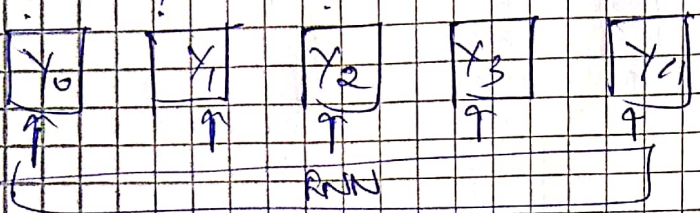
** final gradient = $\nabla_{Y_t} DIV = [0 \quad 0 \quad \ldots \quad \frac{-1}{Y(t, \text{symbol}_t)} \quad 0 \ldots 0]$

(zeros at all location/symbols except locations of span of target)

(partial derivative for each symbol results in being treating other sym as constant so after differentiate we get zero for other symbol)

What happens if no Timing info is provided?

/B/ /I/ /I/ /E/ /I/ /E/



Only we know o/p labels or sequence but don't know when it occurs

How do we compute divergence?

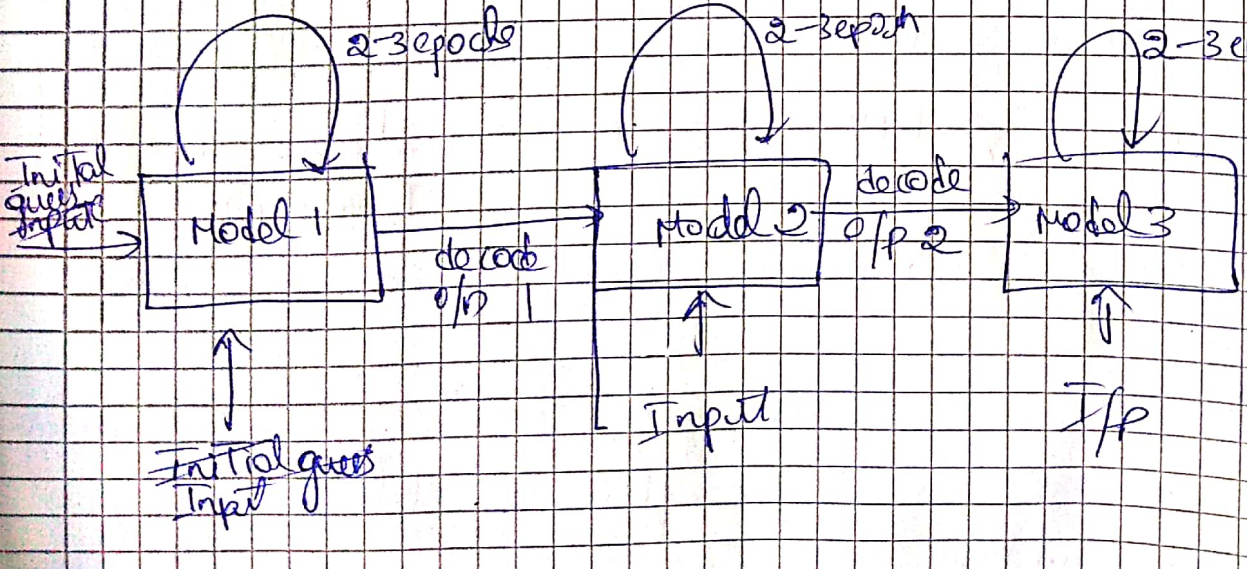
Solution 1 :- Guess the alignment at start

Initialization :- Guess the alignment randomly / manually / let

Training :- Pass the ipputs get the o/p & find gradient

Learning :- Use the output alignment Train network for 2-3 epochs for current

Iterate :- Decode trained n/w o/p & provide it as input for next new n/w. Again retrain for 2-3 epochs & iterate



Estimating alignment

a) Given :- → Unaligned K length symbol sequence $S = S_0 \ldots S_{K-1}$ (e.g: /B/ /I/ /F/ /I/)

→ An N-length input ($N \geq K$)

→ And a trained recurrent n/w

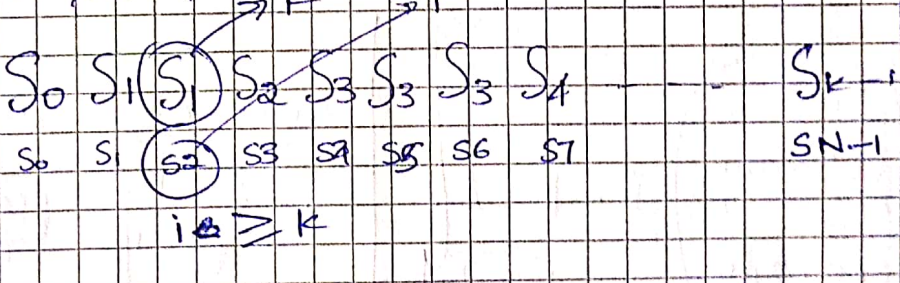
b) Find (By repeating symbols final sequence length = Input sequence length)

→ An N length expansion $S_0 \ldots S_{N-1}$ comprising of symbols in S in start order eg: $S_0 S_1 S_1 S_2 S_3 S_3 S_3 \ldots S_{K-1}$

→ $S_i = S_K \Rightarrow i \geq K$

→ $S_i = S_K$, $S_j = S_L$, $i < j \Rightarrow K \leq L$

c) Outcome:- An alignment of tgt symbol sequence $S_0 \ldots S_{K-1}$ To input $X_0 \ldots X_{N-1}$



→ We would use viterbi algorithm

Unconstrained decoding

→ Actual op of network at each timestep is simply probability distro over all symbols

→ We can condition on input & get most likely symbols

→ Output may not correspond To legal sequence

Problems : Symbols in unconstrained o/p may not be o/p symbol required

Solution : Block out all output symbols not there in target sequence

→ Even after this we may get diffrent sequence

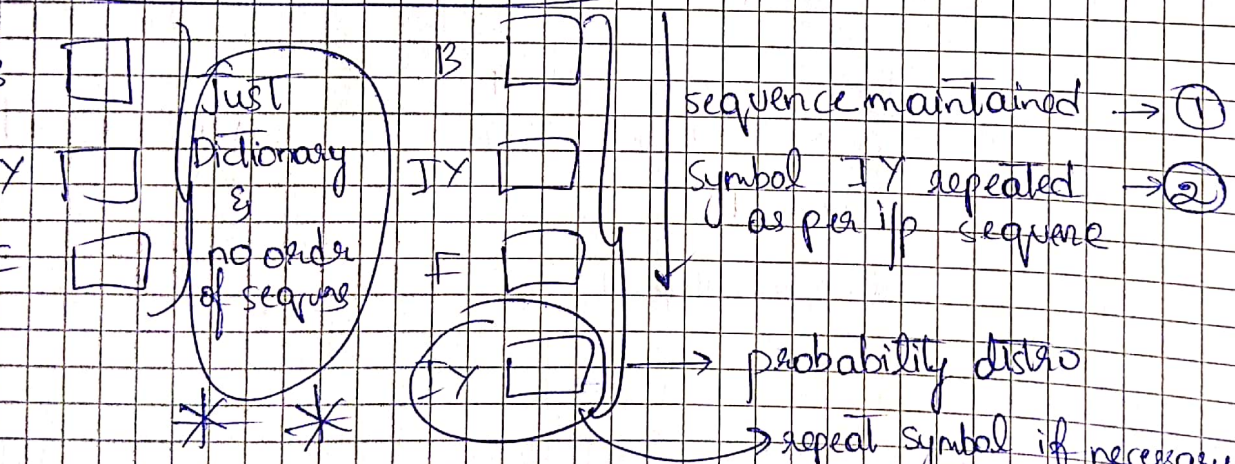
Input: BIYFIY

Output without constraints : (B) IY B F (M) → Symbols not in dictionary

Output with dictionary : B F B IY B Incorrect sequence
soln 1 constraint

Soln 2 : Symbols should be in dictionary and also sequence must be the

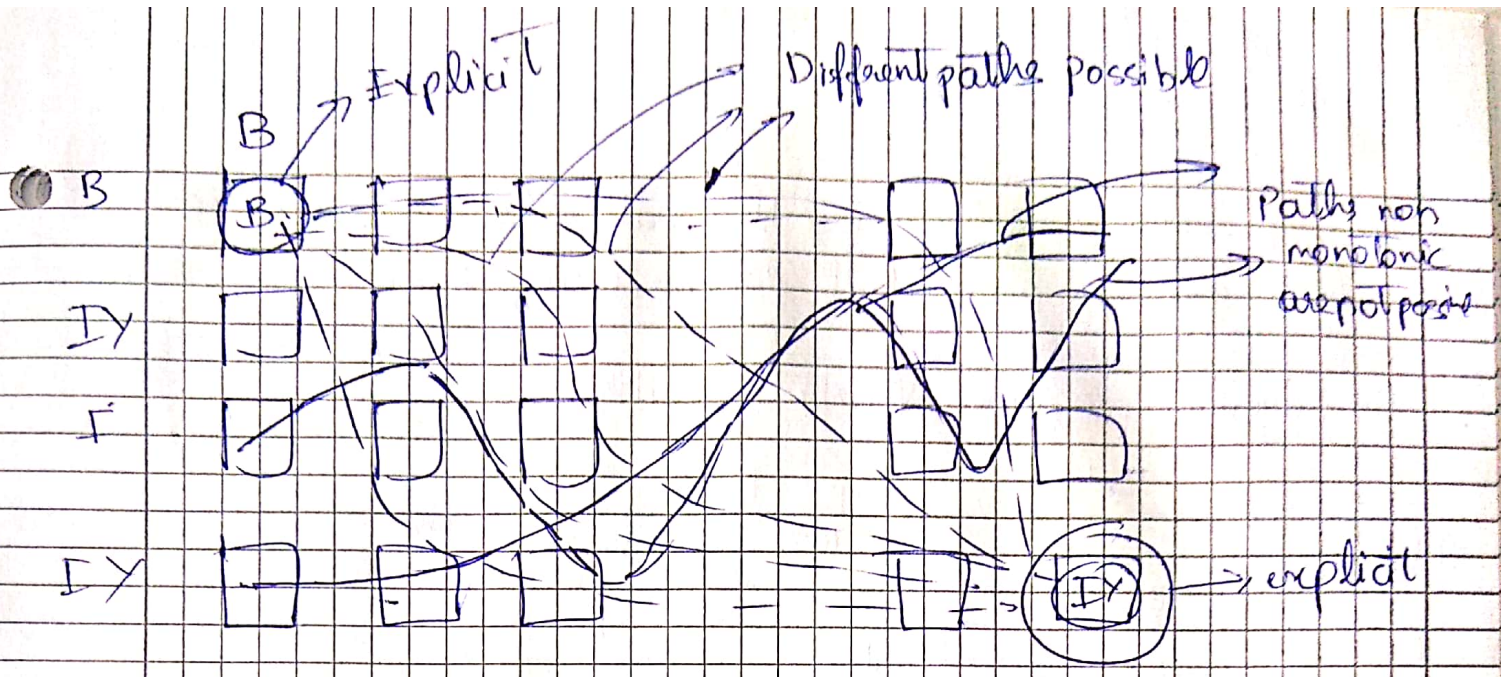
soln 1 V/S soln 2 * *



Explicit constraint on alignment

→ First symbol must be in top left block when decoded

→ Last symbol at bottom right block during decoding



→ Only Monotonic Traversal (dotted line), non-monotonic (solid line) not possible

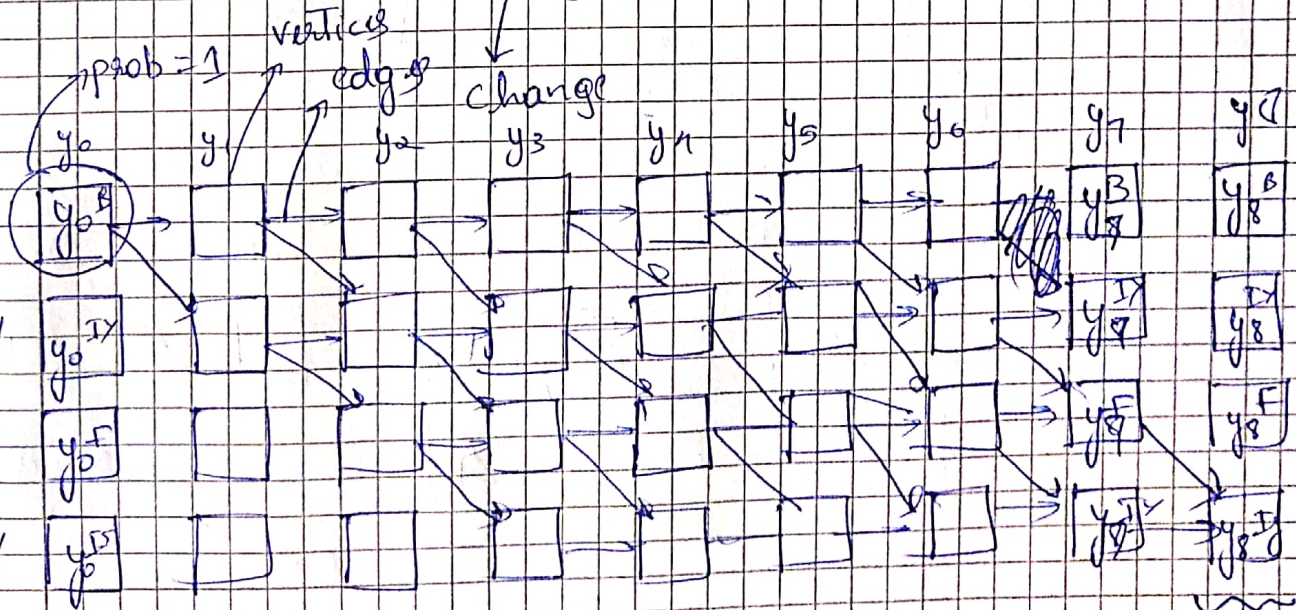
** → Monotonicity assures final o/p has some order maintained even if it fails to get some symbols in between

Example :

B	B	IY	F	IY
B	IY	IY	IY	
B	F	IY	IY	

possible sequences

constant



** → Vertices contain probability, edges value hard coded

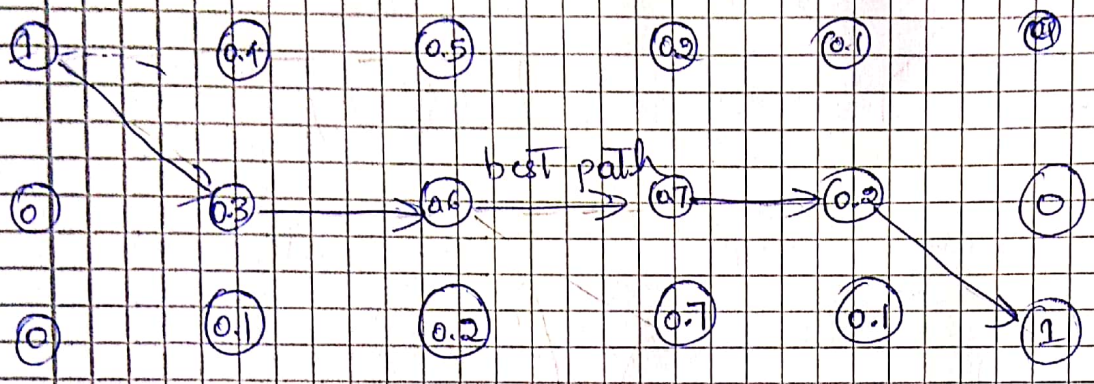
To (1)

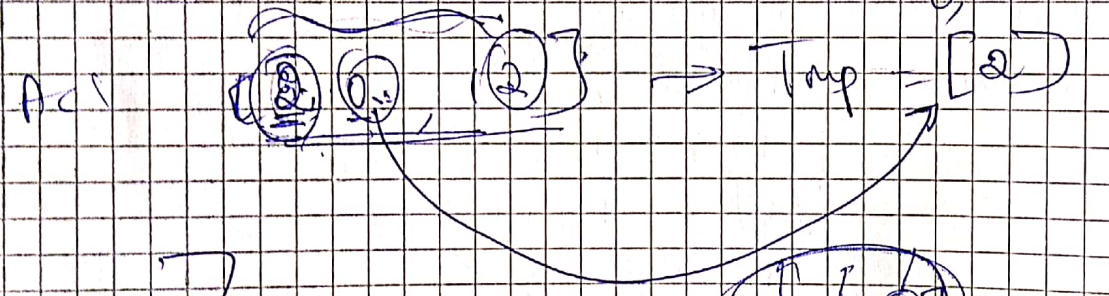
**

→ Most probable sequence is the path having max score where score is product of probabilities from B → IY

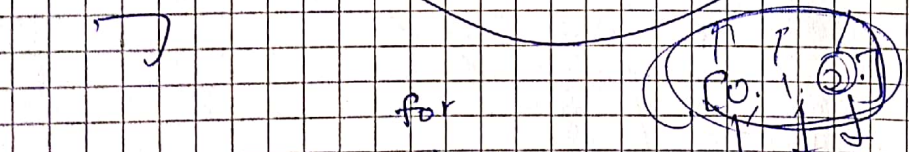
*

→ Use dynamic program Viterbi algorithm





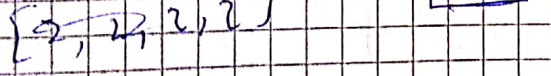
max

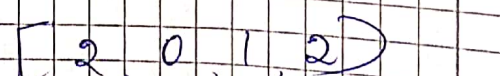


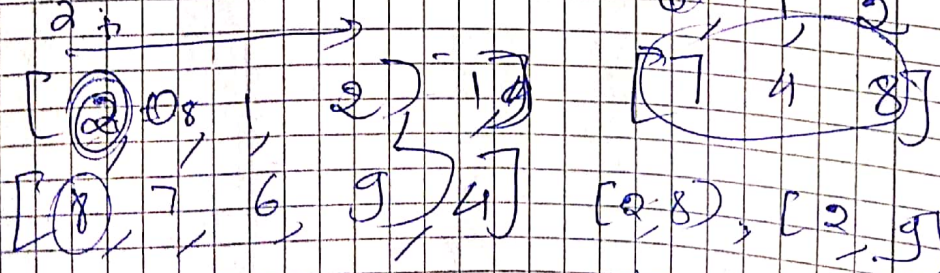
for

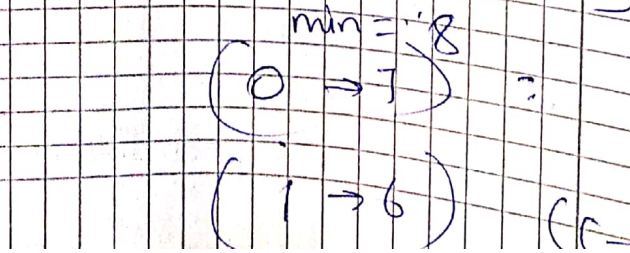
temp[2]

bid se



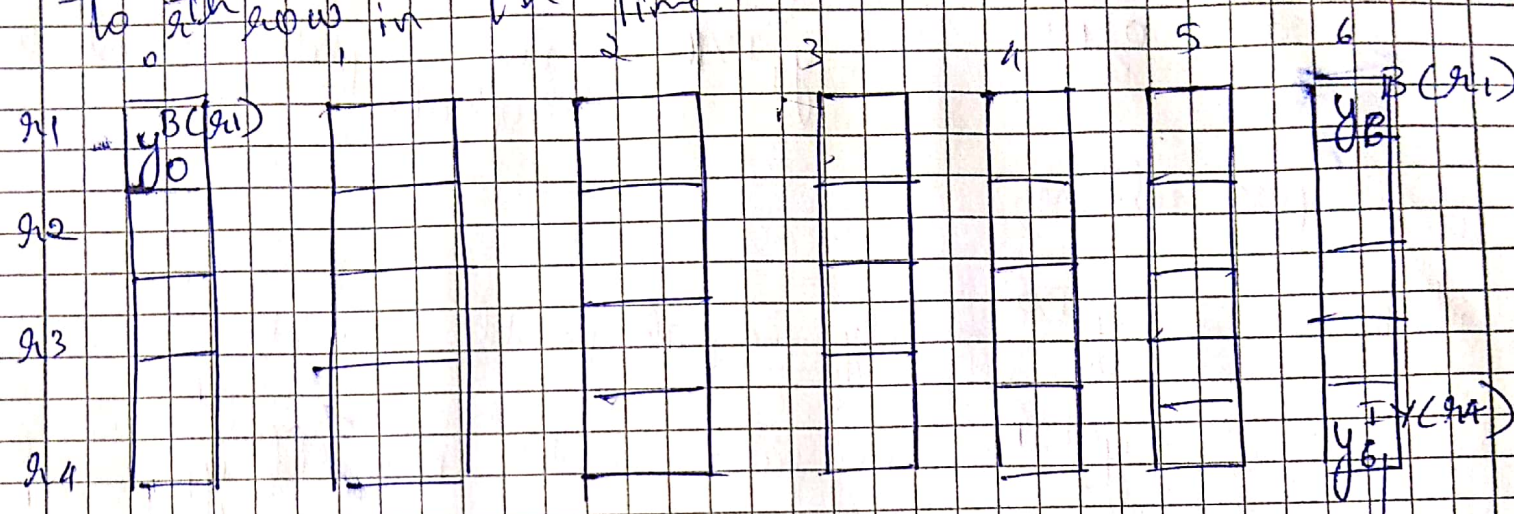






Viterbi Algorithm

Notation: let $y_i^{s(q)}$ is prob of i^{th} symbol assigned to q^{th} row in i^{th} time



Initialization:

1st Time Step:

No predecessor $\Rightarrow BP(0,i) = null \quad i = 0 \to K-1$

$Bscr(0,0) = y_0^{s(0)}$, $Bscr(0,i) = -\infty$, $i = 1 \to K-1$

for $t = 1 \to T-1$

$BP(t,0) = 0$; $Bscr(t,0) = Bscr(t-1,0) * y_t^{s(0)}$

for $l = 1 \to K-1$

$\rightarrow BP(t,l) = (\text{if } (Bscr(t-1,l) > Bscr(t-1,l-1), l, l-1), \text{ else } l)$

$Bscr(t,l) = Bscr(BP(t,l)) * y_t^{s(l)}$

Timestep 0 :- No predecessor

$scor = y_0^B$

Timestep (1) :-

raws $\{0, 1\}$

y_1^{BO} → only one path possible hence use it

y_1^{IY} → Only one path possible hence update

Compare whether

Timestep (2) :- y_2^{BO} → only one path possible use it

y_2^{IY} → Two paths possible now if $y_1^{IY} > y_1^B$ choose y_1^{IY} as predecessor

y_2^F → Only one path y_1^{IY}

T3:- $y_3^{BO} \rightarrow y_2^{BO}$

$y_3^{IY} \rightarrow$ predecessor

$y_2^B * y_3^{IY}$ if $y_2^B > y_2^{IY}$

$y_2^{IY} * y_3^{IY}$ else

$y_3^F \rightarrow$ predecessor

$y_2^{IY} * y_3^F$ if $y_2^{IY} > y_2^F$

$y_2^F * y_3^F$ else

$y_3^{IY} \rightarrow y_2^F$

Greedy v/s Beam

→ Greedy can pick only one symbol at each time step, helpful if one/two symbols have much higher prob than others

→ Beam can help with generating multiple alignments for a single o/p

B ∈ IY ∈ FIY
B B ∈ IY F IY
B ∈ ∈ ∈ IY F IY

→ But not guaranteed to find most likely Y

Beam search $\frac{1}{T_y^{\alpha}} \sum_{t=1}^{T_y} \log P(y^{<t>} | x, y^{<1>}, \ldots, y^{<t-1>})$

$\alpha = 0.7$

6/06/20 Stober - CTC

$P(Y|X)$ = Sum of all valid alignments → dynamic programming

RNN output = $\prod_{t=1}^{T} P_t(a_t | X)$ → ~~loss~~ each alignment probability

→ should be differentiable

→ Since we don't know exact time position where each character occurs, we need to aggregate the each individual (valid alignments) & treat them all as one

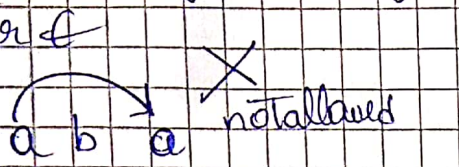
B I Y ∈ F ∈ Y
B ∈ I Y F ∈ Y
B ∈ I Y ∈ F ∈ Y
B B I Y ∈ F F ∈ Y Y

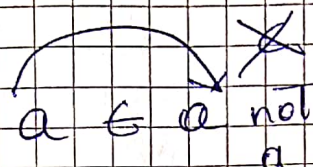
} all are valid

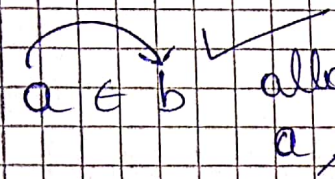
Allowed Transitions v/s not allowed transition

Introduce ϵ To differentiate b/w Two repeated characters

Collapse every thing / symbol repeated To unique symbol before & after ϵ







Greedy

→ Doesn't consider multiplier alignment

** MAP v/s MLE hypothesis

Sum of prob belonging To one class should be greater

greedy

Example	prob
b a b	0.05
a a ϵ	0.3
a ϵ a	0.05
b b ~~b~~	0.4
ϵ a a	0.1

a a ϵ } greedy
a a a } → both beam

If we consider greedy Then most probable alignment v/s is b b ~~b~~ with prob of 0.4 It like MLE hypothesis

But realistically P (a a ϵ | a ϵ a | ϵ a a) = 0.1 + 0.05 + 0.3 = 0.45

hence a a ϵ is required sequence which like MAP

Beam Search Heuristic

Applying Language Model

→ Just beam search / greedy results in output at each time step being generated conditionally independent of its previous time step

→ In real world this is different, Use some Language prior while calculating Combined prob

$$y^* = \underset{y}{\operatorname{argmax}} \underbrace{P(Y/X)}_{\text{CTC conditional prob}} \cdot \underbrace{P(Y)^{\alpha}}_{\text{language model prob}} \cdot \underbrace{L(Y)^{\beta}}_{\text{Word insertion bonus}}$$

** → Normally used during deployment / decoding & not during RNN Training

** → $P(Y)^{\alpha}$ → can be trained independently on Wikipedia Text to know general structure of language (Domain (law/medical))

* → $L(Y)^{\beta}$ → Helps to remove bias on short Target sequence

$$\log \operatorname{argmax} P(Y/X) + \alpha \log P(Y) + \beta \log L(Y)$$

→ $P(Y)^{\alpha}$ → can be swapped b/wn domains

HMM

Monotonicity → cannot go back, skip over blank state