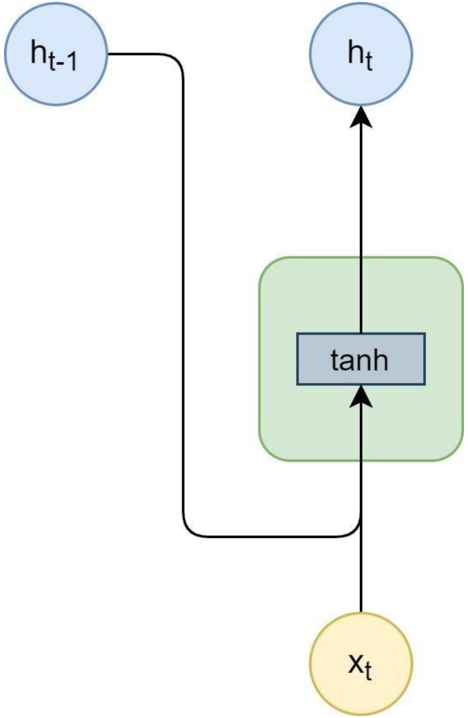
HW3P1 Bootcamp

RNN, GRU, CTC, and Greedy/Beam Search
(Spring 2023)

Harshith Arun Kumar

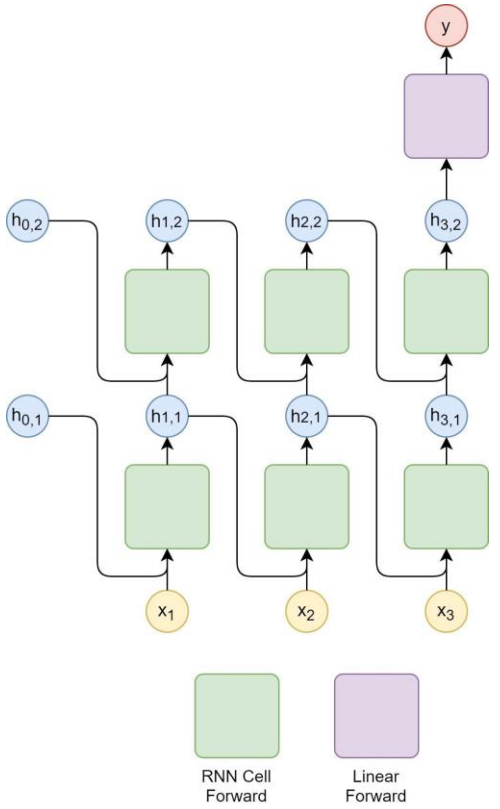
RNN Cell Forward / Backward

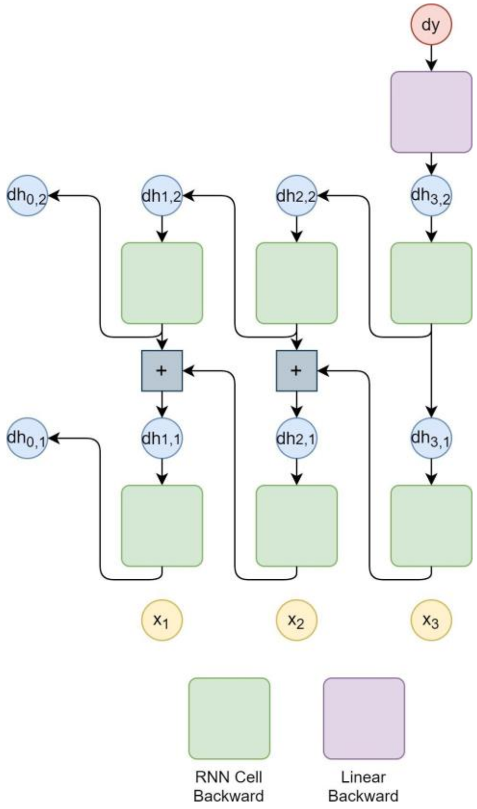


$$h_t = tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh})$$

Tip: Very similar to how you did ***linear.py*** in hw1p1.

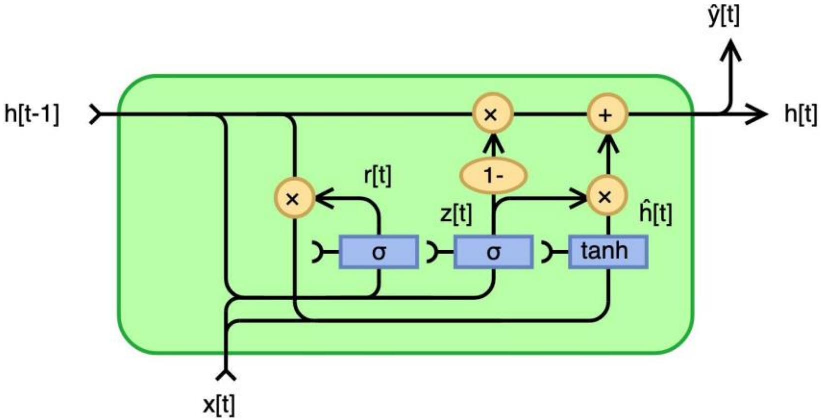
RNN Phoneme Classifier





- Forward-> Follow the TODO
- Backward is tricky
    - 2 diagrams in the write up for understanding the data flow
    - Then follow the pseudocode exactly

GRU Cell Forward/Backward



$$\mathbf{r}_t = \sigma(\mathbf{W}_{ir}\mathbf{x}_t + \mathbf{b}_{ir} + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_{hr})$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_{iz}\mathbf{x}_t + \mathbf{b}_{iz} + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_{hz})$$

$$\mathbf{n}_t = \tanh(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{b}_{in} + \mathbf{r}_t \otimes (\mathbf{W}_{hn}\mathbf{h}_{t-1} + \mathbf{b}_{hn}))$$

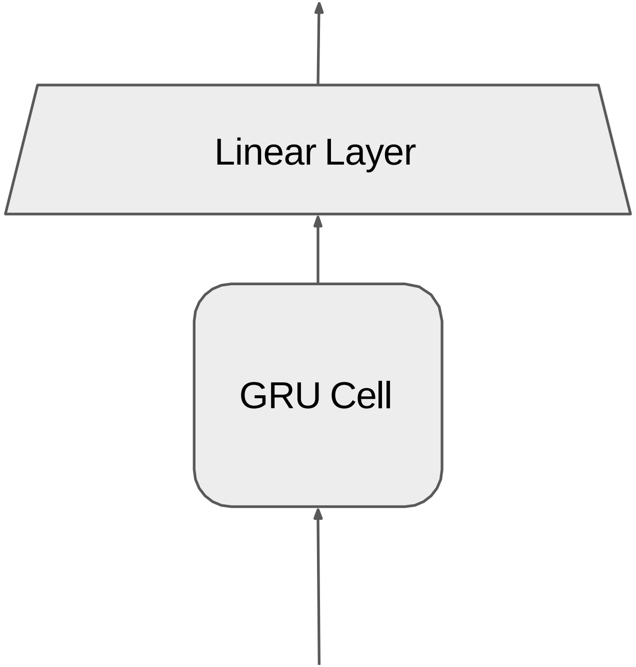
$$\mathbf{h}_t = (1 - \mathbf{z}_t) \otimes \mathbf{n}_t + \mathbf{z}_t \otimes \mathbf{h}_{t-1}$$

https://colah.github.io/posts/2015-08-Backprop

GRU Cell Forward / Backward

- GRU backward be the longest question in HW3P1
- Tips:
  - Modify the ***test_gru.py*** code accordingly – all **dWs** and **dbs** should correct to make sure that your **dx** and **dh** are correct
  - Can try to decompose eqns in forward (That’s how I did :’))
    - A = Tanh(Wx*x + bx + Wh*h + bh)
      - Z1 = Wx*x + bx
      - Z2 = Wh*h + bh
      - Z = Z1 + Z2
      - A = Tanh(Z)
    - Backward is relatively easy now. Need to compute the gradients in this order. Given dA (actually dLdA – ignoring for simplicity)
      - dZ -> dZ1, dZ2 -> dWh, dh, dbh -> …

GRU Inference

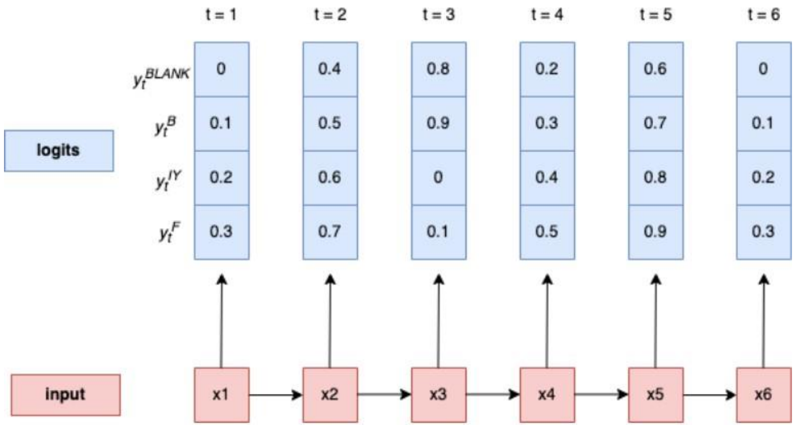


CTC based questions

- Lecture slides have everything needed to complete all the CTC sections and also decoding

CTC based questions

- We have given example questions for you to understand the math behind it

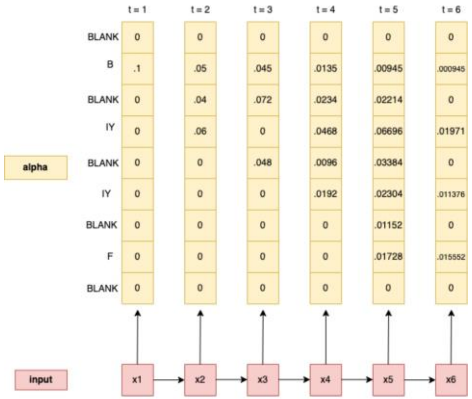


	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6
y_t^{BLANK}	0	0.4	0.8	0.2	0.6	0
y_t^{B}	0.1	0.5	0.9	0.3	0.7	0.1
y_t^{IY}	0.2	0.6	0	0.4	0.8	0.2
y_t^{F}	0.3	0.7	0.1	0.5	0.9	0.3

Figure 12: An overall CTC setup example

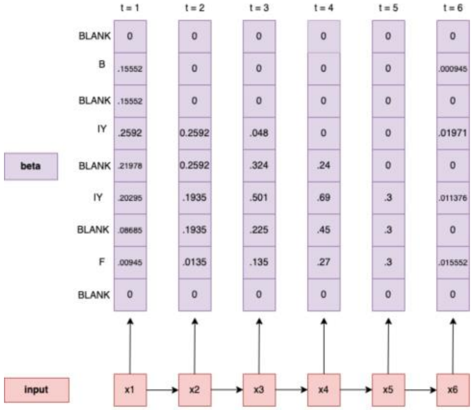
CTC based questions

- We have given example questions for you to understand the math behind it



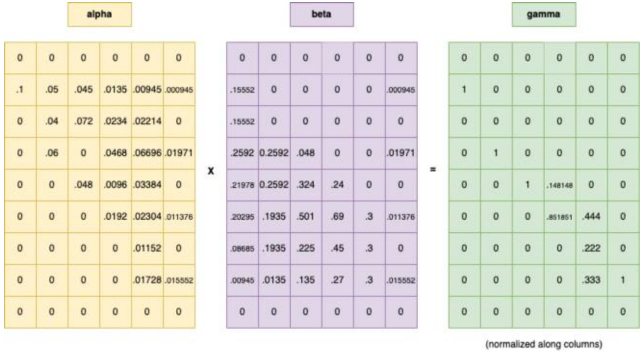
	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6
BLANK	0	0	0	0	0	0
B	.1	.05	.045	.0135	.00945	.000945
BLANK	0	.04	.072	.0234	.02214	0
IY	0	.06	0	.0468	.06696	.01971
BLANK	0	0	.048	.0096	.03384	0
IY	0	0	0	.0192	.02304	.011376
BLANK	0	0	0	0	.01152	0
F	0	0	0	0	.01728	.015552
BLANK	0	0	0	0	0	0

Figure 15: Forward Algorithm



	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6
BLANK	0	0	0	0	0	0
B	.15552	0	0	0	0	.000945
BLANK	.15552	0	0	0	0	0
IY	.2592	0.2592	.048	0	0	.01971
BLANK	.21978	0.2592	.324	.24	0	0
IY	.20295	.1935	.501	.69	.3	.011376
BLANK	.08685	.1935	.225	.45	.3	0
F	.00945	.0135	.135	.27	.3	.015552
BLANK	0	0	0	0	0	0

Figure 16: Backward Algorithm



alpha

0	0	0	0	0	0
.1	.05	.045	.0135	.00945	.000945
0	.04	.072	.0234	.02214	0
0	.06	0	.0468	.06696	.01971
0	0	.048	.0096	.03384	0
0	0	0	.0192	.02304	.011376
0	0	0	0	.01152	0
0	0	0	0	.01728	.015552
0	0	0	0	0	0

x

beta

0	0	0	0	0	0
.15552	0	0	0	0	.000945
.15552	0	0	0	0	0
.2592	0.2592	.048	0	0	.01971
.21978	0.2592	.324	.24	0	0
.20295	.1935	.501	.69	.3	.011376
.08685	.1935	.225	.45	.3	0
.00945	.0135	.135	.27	.3	.015552
0	0	0	0	0	0

=

gamma

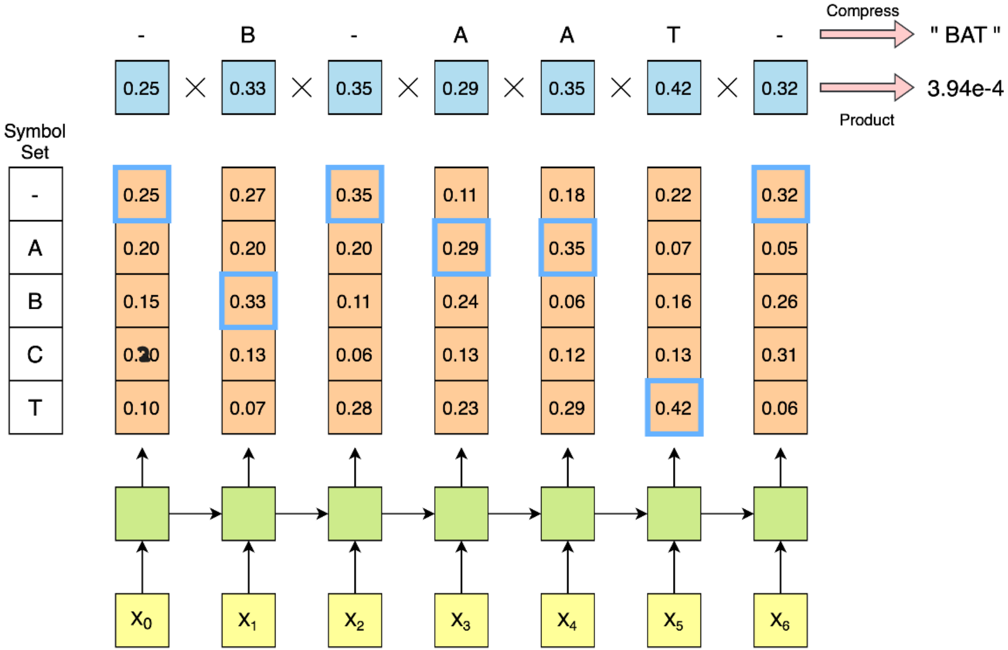
0	0	0	0	0	0
1	0	0	0	0	0
0	0	0	0	0	0
0	1	0	0	0	0
0	0	1	.148148	0	0
0	0	0	.851851	.444	0
0	0	0	0	.222	0
0	0	0	0	.333	1
0	0	0	0	0	0

(normalized along columns)

Figure 17: Posterior Probability

Greedy Search

- Taking the most probably output at each time step

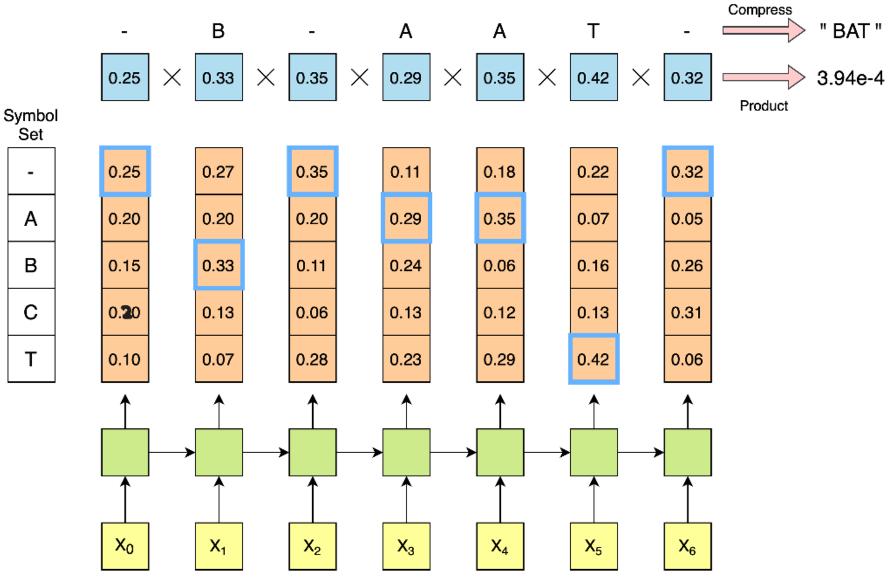


	-	B	-	A	A	T	-
	0.25	0.33	0.35	0.29	0.35	0.42	0.32

Symbol Set	X_0	X_1	X_2	X_3	X_4	X_5	X_6
-	0.25	0.27	0.35	0.11	0.18	0.22	0.32
A	0.20	0.20	0.20	0.29	0.35	0.07	0.05
B	0.15	0.33	0.11	0.24	0.06	0.16	0.26
C	0.30	0.13	0.06	0.13	0.12	0.13	0.31
T	0.10	0.07	0.28	0.23	0.29	0.42	0.06

Greedy Search

- Taking the most probably output at each time step



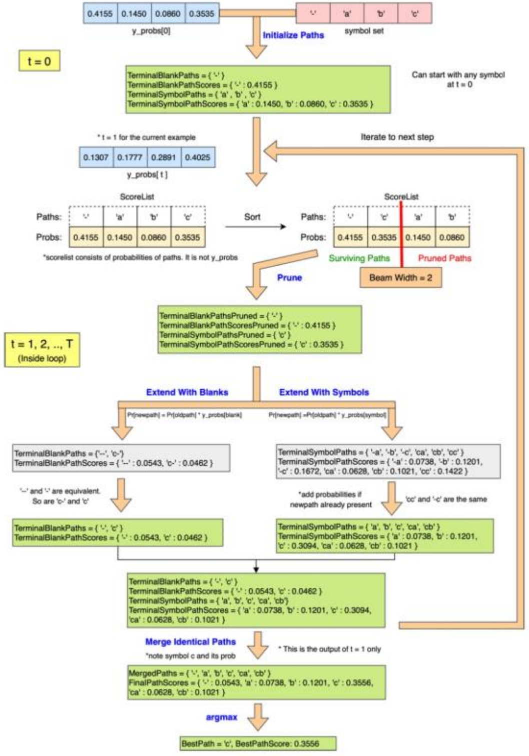
Symbol Set							
-	0.25	0.27	0.35	0.11	0.18	0.22	0.32
A	0.20	0.20	0.20	0.29	0.35	0.07	0.05
B	0.15	0.33	0.11	0.24	0.06	0.16	0.26
C	0.30	0.13	0.06	0.13	0.12	0.13	0.31
T	0.10	0.07	0.28	0.23	0.29	0.42	0.06

Write your compress function separately

Can complete without a for loop but a for loop wont cause autolab to time out

Beam Search

- Another hard question in this part
- Tips to complete this question fast
  - Understand beam search from the lecture videos and slides
  - Beware of the definition of ***set()*** *(python { })* and ***list()*** *(python [ ])* from the code given in lecture slides. There is a difference in the python implementation
  - Complete each function ***InitializePaths, Prune, ExtendWithBlank, ExtendWithSymbol, MergeIdenticalPaths*** individually and then check your outputs with the flow chart given in the write up


0.4155
0.1450
0.0860
0.3535
y_probs[0]
'-'
'a'
'b'
'c'
symbol set
Initialize Paths
t = 0
Can start with any symbol at t = 0
* t = 1 for the current example
0.1307
0.1777
0.2891
0.4025
y_probs[t]
Iterate to next step
ScoreList
Paths:
'-'
'a'
'b'
'c'
Probs:
0.4155
0.1450
0.0860
0.3535
*scorelist consists of probabilities of paths. It is not y_probs
Sort
ScoreList
Paths:
'-'
'c'
'a'
'b'
Probs:
0.4155
0.3535
0.1450
0.0860
Surviving Paths
Pruned Paths
Beam Width = 2
Prune
t = 1, 2, .., T
(Inside loop)
Extend With Blanks
Extend With Symbols
Pr[newpath] = Pr[oldpath] * y_probs[blank]
Pr[newpath] =Pr[oldpath] * y_probs[symbol]
'--' and '-' are equivalent. So are 'c-' and 'c'
*add probabilities if newpath already present
'cc' and '-c' are the same
Merge Identical Paths
*note symbol c and its prob
* This is the output of t = 1 only
argmax

```
TerminalBlankPaths = { '-' }
TerminalBlankPathScores = { '-' : 0.4155 }
TerminalSymbolPaths = { 'a' , 'b' , 'c' }
TerminalSymbolPathScores = { 'a' : 0.1450, 'b' : 0.0860, 'c' : 0.3535 }
```

```
TerminalBlankPathsPruned = { '-' }
TerminalBlankPathScoresPruned = { '-' : 0.4155 }
TerminalSymbolPathsPruned = { 'c' }
TerminalSymbolPathScoresPruned = { 'c' : 0.3535 }
```

```
TerminalBlankPaths = {'--', 'c-'}
TerminalBlankPathScores = { '--' : 0.0543, 'c-' : 0.0462 }
```

```
TerminalSymbolPaths = { '-a', '-b', '-c', 'ca', 'cb', 'cc' }
TerminalSymbolPathScores = { '-a' : 0.0738, '-b' : 0.1201,
'-c' : 0.1672, 'ca' : 0.0628, 'cb' : 0.1021, 'cc' : 0.1422 }
```

```
TerminalBlankPaths = { '-', 'c' }
TerminalBlankPathScores = { '-' : 0.0543, 'c' : 0.0462 }
```

```
TerminalSymbolPaths = { 'a', 'b', 'c', 'ca', 'cb' }
TerminalSymbolPathScores = { 'a' : 0.0738, 'b' : 0.1201,
'c' : 0.3094, 'ca' : 0.0628, 'cb' : 0.1021 }
```

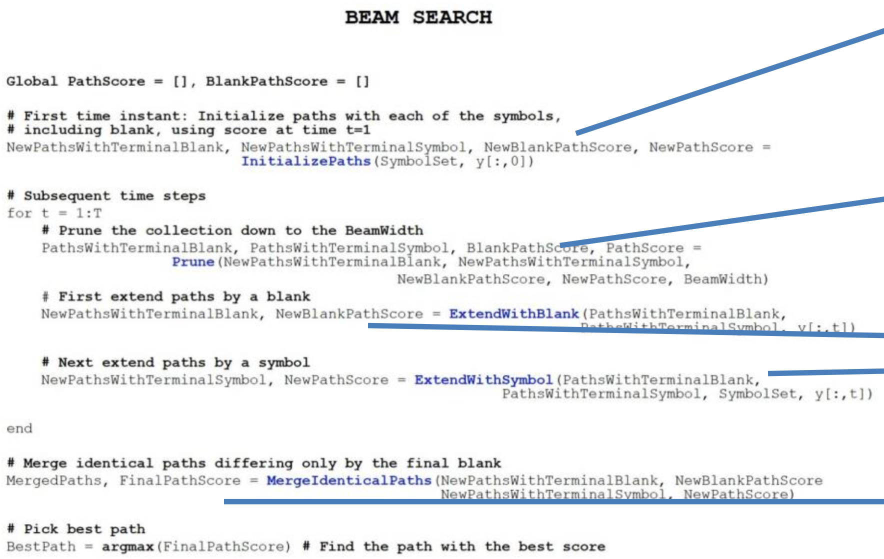
```
TerminalBlankPaths = { '-', 'c' }
TerminalBlankPathScores = { '-' : 0.0543, 'c' : 0.0462 }
TerminalSymbolPaths = { 'a', 'b', 'c', 'ca', 'cb'}
TerminalSymbolPathScores = { 'a' : 0.0738, 'b' : 0.1201, 'c' : 0.3094,
'ca' : 0.0628, 'cb' : 0.1021 }
```

```
MergedPaths = { '-', 'a', 'b', 'c', 'ca', 'cb' }
FinalPathScores = { '-' : 0.0543, 'a' : 0.0738, 'b' : 0.1201, 'c' : 0.3556,
'ca' : 0.0628, 'cb' : 0.1021 }
```

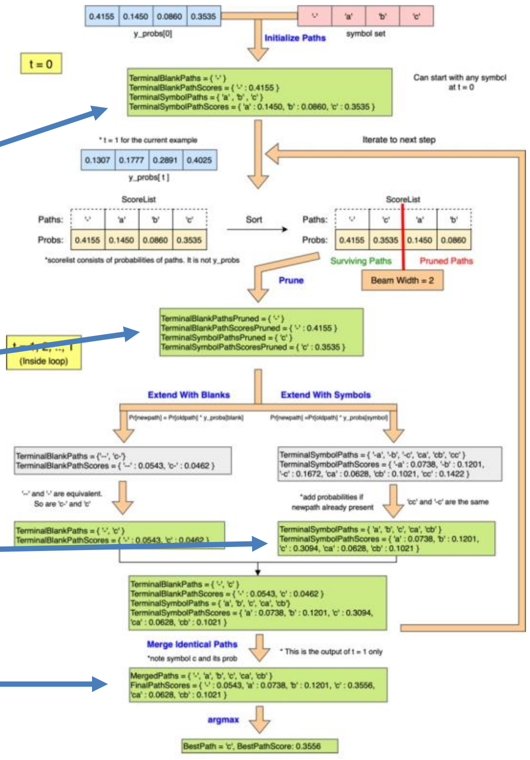
```
BestPath = 'c', BestPathScore: 0.3556
```

Figure 19: Beam Search

Beam Search



```
                         BEAM SEARCH

Global PathScore = [], BlankPathScore = []

# First time instant: Initialize paths with each of the symbols,
# including blank, using score at time t=1
NewPathsWithTerminalBlank, NewPathsWithTerminalSymbol, NewBlankPathScore, NewPathScore =
                           InitializePaths(SymbolSet, y[:,0])

# Subsequent time steps
for t = 1:T
    # Prune the collection down to the BeamWidth
    PathsWithTerminalBlank, PathsWithTerminalSymbol, BlankPathScore, PathScore =
                   Prune(NewPathsWithTerminalBlank, NewPathsWithTerminalSymbol,
                                            NewBlankPathScore, NewPathScore, BeamWidth)
    # First extend paths by a blank
    NewPathsWithTerminalBlank, NewBlankPathScore = ExtendWithBlank(PathsWithTerminalBlank,
                                                                PathsWithTerminalSymbol, y[:,t])

    # Next extend paths by a symbol
    NewPathsWithTerminalSymbol, NewPathScore = ExtendWithSymbol(PathsWithTerminalBlank,
                                                  PathsWithTerminalSymbol, SymbolSet, y[:,t])

end

# Merge identical paths differing only by the final blank
MergedPaths, FinalPathScore = MergeIdenticalPaths(NewPathsWithTerminalBlank, NewBlankPathScore
                                                  NewPathsWithTerminalSymbol, NewPathScore)

# Pick best path
BestPath = argmax(FinalPathScore) # Find the path with the best score
```



ScoreList				
Paths:	'-'	'a'	'b'	'c'
Probs:	0.4155	0.1450	0.0860	0.3535

*scorelist consists of probabilities of paths. It is not y_probs

ScoreList				
Paths:	'-'	'c'	'a'	'b'
Probs:	0.4155	0.3535	0.1450	0.0860

TerminalBlankPaths = { '-' }
TerminalBlankPathScores = { '-' : 0.4155 }
TerminalSymbolPaths = { 'a' , 'b' , 'c' }
TerminalSymbolPathScores = { 'a' : 0.1450, 'b' : 0.0860, 'c' : 0.3535 }

TerminalBlankPathsPruned = { '-' }
TerminalBlankPathScoresPruned = { '-' : 0.4155 }
TerminalSymbolPathsPruned = { 'c' }
TerminalSymbolPathScoresPruned = { 'c' : 0.3535 }

TerminalBlankPaths = {'--', 'c-'}
TerminalBlankPathScores = { '--' : 0.0543, 'c-' : 0.0462 }

'--' and '-' are equivalent.
So are 'c-' and 'c'

TerminalBlankPaths = { '-', 'c' }
TerminalBlankPathScores = { '-' : 0.0543, 'c' : 0.0462 }

TerminalSymbolPaths = { '-a', '-b', '-c', 'ca', 'cb', 'cc' }
TerminalSymbolPathScores = { '-a' : 0.0738, '-b' : 0.1201,
'-c' : 0.1672, 'ca' : 0.0628, 'cb' : 0.1021, 'cc' : 0.1422 }

*add probabilities if newpath already present

'cc' and '-c' are the same

TerminalSymbolPaths = { 'a', 'b', 'c', 'ca', 'cb' }
TerminalSymbolPathScores = { 'a' : 0.0738, 'b' : 0.1201,
'c' : 0.3094, 'ca' : 0.0628, 'cb' : 0.1021 }

TerminalBlankPaths = { '-', 'c' }
TerminalBlankPathScores = { '-' : 0.0543, 'c' : 0.0462 }
TerminalSymbolPaths = { 'a', 'b', 'c', 'ca', 'cb'}
TerminalSymbolPathScores = { 'a' : 0.0738, 'b' : 0.1201, 'c' : 0.3094,
'ca' : 0.0628, 'cb' : 0.1021 }

*note symbol c and its prob

* This is the output of t = 1 only

MergedPaths = { '-', 'a', 'b', 'c', 'ca', 'cb' }
FinalPathScores = { '-' : 0.0543, 'a' : 0.0738, 'b' : 0.1201, 'c' : 0.3556,
'ca' : 0.0628, 'cb' : 0.1021 }

BestPath = 'c', BestPathScore: 0.3556

Figure 19: Beam Search

- Green boxes show the output for the 1st test case in the local autograder for just 1 time step
- You can break the flow in-between and check your answers

Thank you!

Q & A