Recitation 9
(CTC Decoding and Beam Search)

Alex, Andy, Puru
Credits: Gabrial, Harini & Quentin

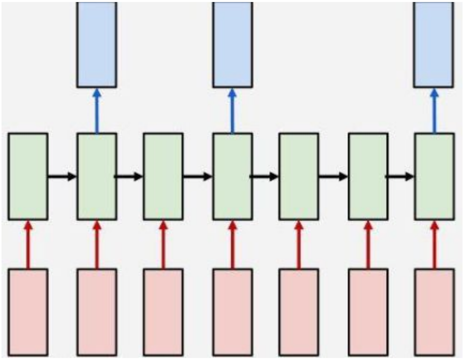
HW1P2 vs. HW3P2

HW1P2

- Sequence Classification for phoneme recognition.
- **Time-synchronous** outputs.

HW3P2

- Sequence to Sequence with **Order Synchrony**
- Training: Using CTC Loss to deal with...
  - The problem of alignment
  - The problem of repetition
- Inference
  - Greedy Search
  - Beam Search



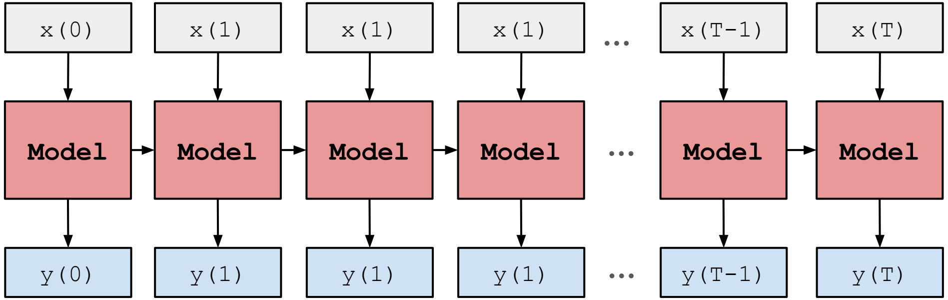
Training

Problem Set-up

- Inputs and targets are not time-aligned
  - $|Y| \neq |X|$
  - $|Y|$ and $|X|$ not proportional
- However, they are order-aligned
- We will compress predicted sequences
- We need to be able to yield repeating outputs after compression

(Training)

Someone says “zoo”



Inputs

Probabilities

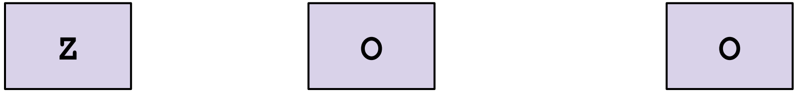
Two problems

(1) time alignment and (2) repetition

One Asset

(1) order alignment

We want to transcribe “zoo” at these time steps



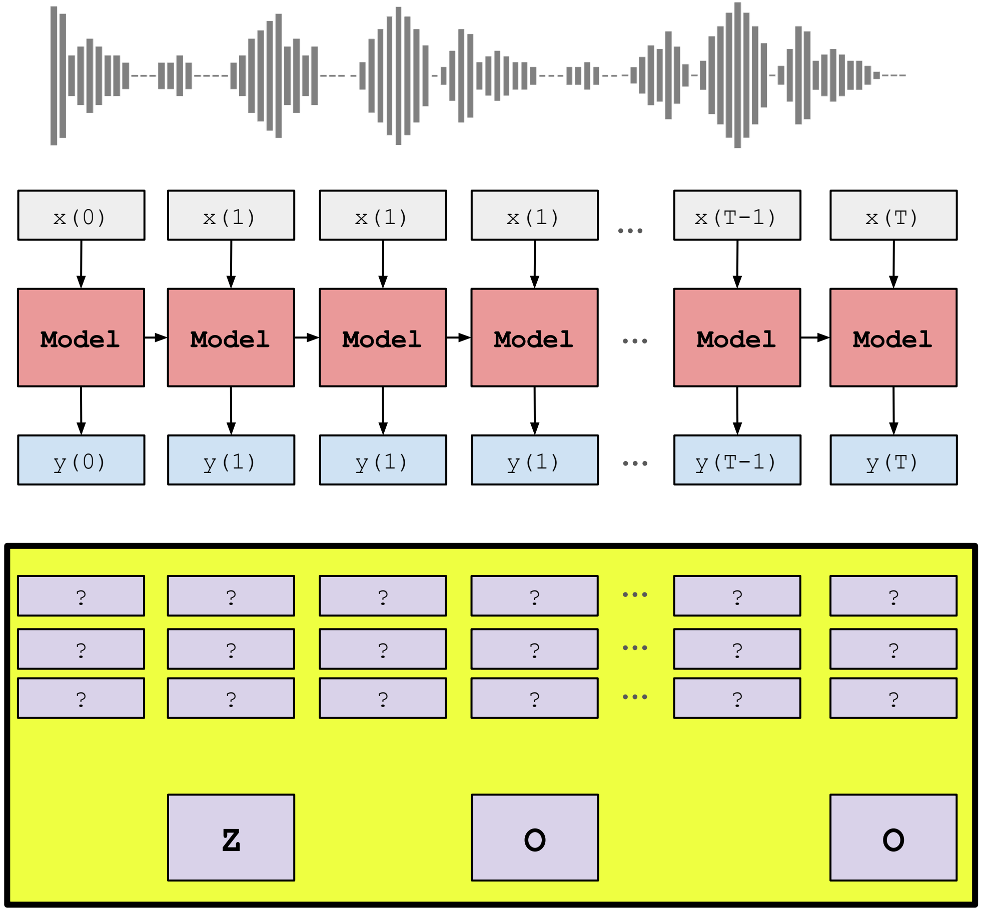
Target

(Training)

Someone says “zoo”

Use **CTC** to get time-aligned targets

We want to transcribe “zoo” at these time steps



Inputs

Probabilities

<u>All</u> time-aligned targets for Target

Target

The problem of repetition…

A note on repetition…

(Training)

"zzzooo"
"zoo"
"zo"
"zzzzzzzzo"

"zo" (🙁)

A note on repetition…

(Training)

"zzzooo"
"zoo"
"zo"
"zzzzzzzzo"

→ "zo" (🙁)

To account for repetitions, **introduce a "break" character ("%")**

"zzz%oo%o"
"zo%o"
~~"zo"~~
~~"zzzzzzzzo"~~

→ "zoo" (😀)

The problem of alignment…

A note on alignment…

(Training)

"z%o%oo"

"z%oo%o"

...

"zooo%o"

➡ "zoo" (🙁)

A note on alignment…

(Training)

“z%o%oo”

“z%oo%o”

…

“zooo%o”

“zoo” (🙁)

If we use a time-aligned target, we can compute a differentiable loss.
But which alignment should we choose?

We *could* use the Viterbi algorithm to find the most probable path and use that to calculate the loss. However, there is a better option…

Using the asset of **order alignment**, as well as some **additional "rules"**, we can actually use **ALL POSSIBLE ALIGNMENTS and calculate an EXPECTED LOSS over them**.

We could use the Viterbi algorithm to find the most probable path and use that to calculate the loss. However, there is a better option...

Using the asset of **order alignment**, as well as some **additional "rules"**, we can actually use **ALL POSSIBLE ALIGNMENTS and calculate an EXPECTED LOSS over them**.

1. Alignments b/t X and Y are **monotonic** (once you advance to the next input, you may only keep the output same advance to the next output)

2. Alignment of X to Y is **many-to-one** (there may be many input elements aligning to a single output element).

3. **Length of Y ≤ Length of X**

We could use the Viterbi algorithm to find the most probable path and use that to calculate the loss. However, there is a better option...

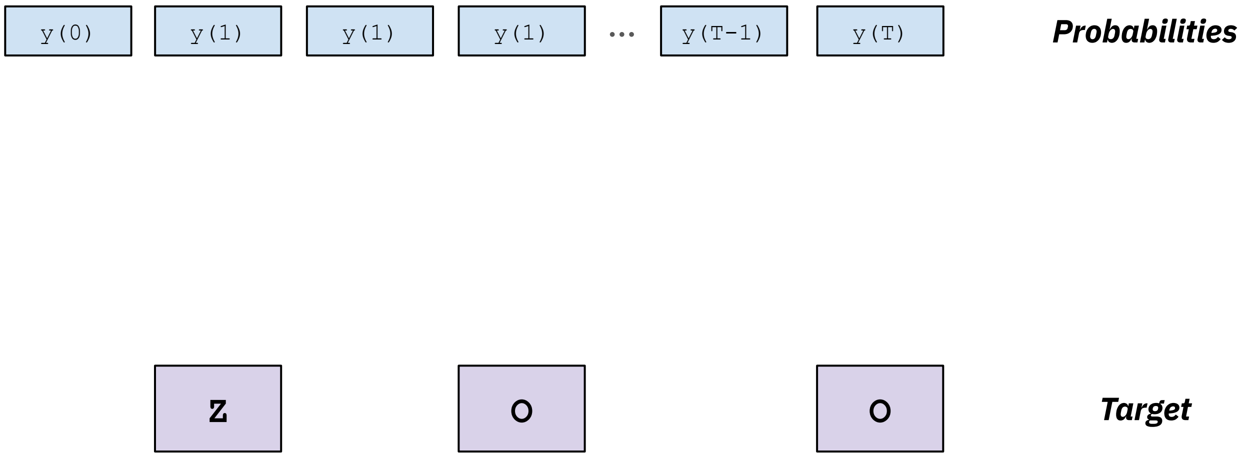
Using the asset of **order alignment**, as well as some **additional "rules"**, we can actually use **ALL POSSIBLE ALIGNMENTS and calculate an EXPECTED LOSS over them**.

⬇

Given: The "break" character, order alignment, and some additional rules…

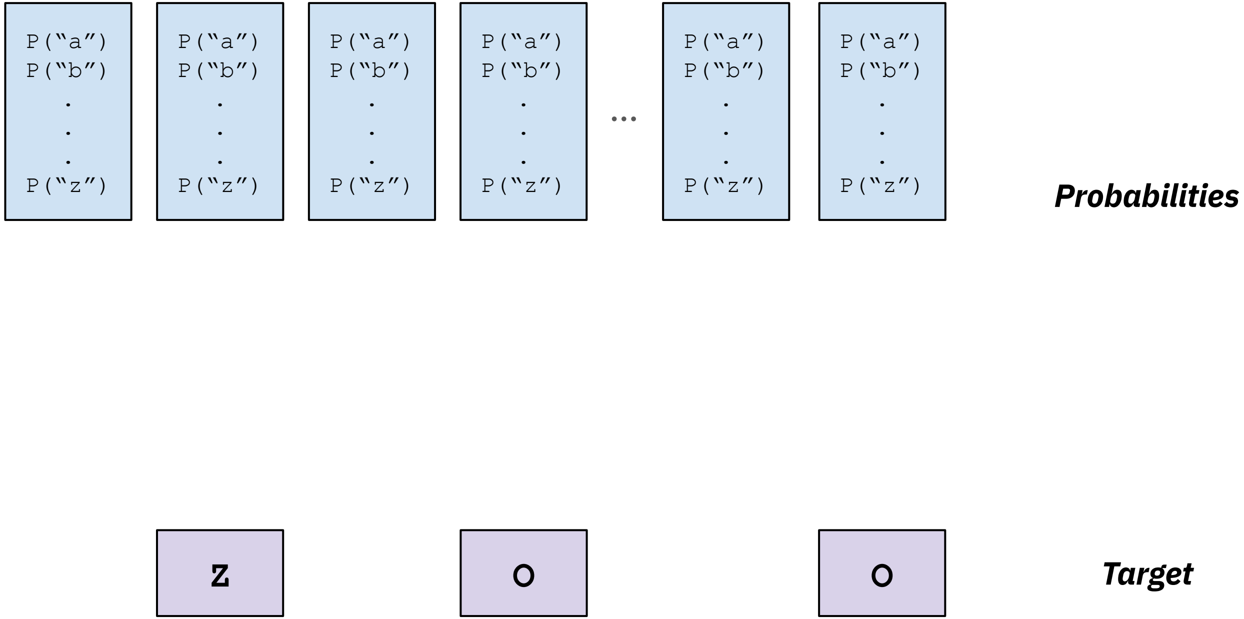
Task: Find possible alignments, their probabilities, then calculate a differentiable loss

(Training)



We want to transcribe "zoo" at these time steps

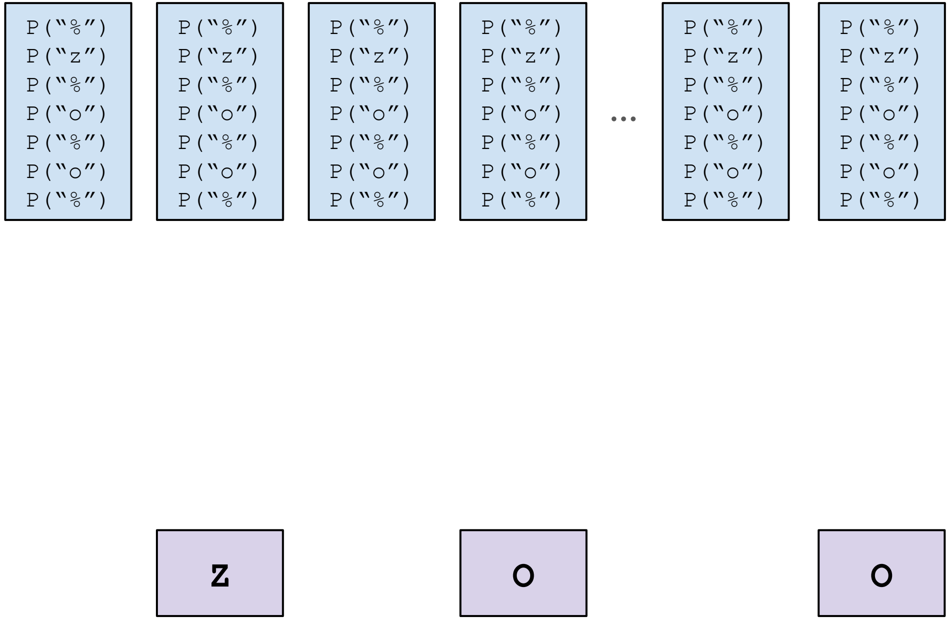
(Training)



We want to transcribe “zoo” at these time steps

(Training)

Limit to the characters in the Target with breaks inserted



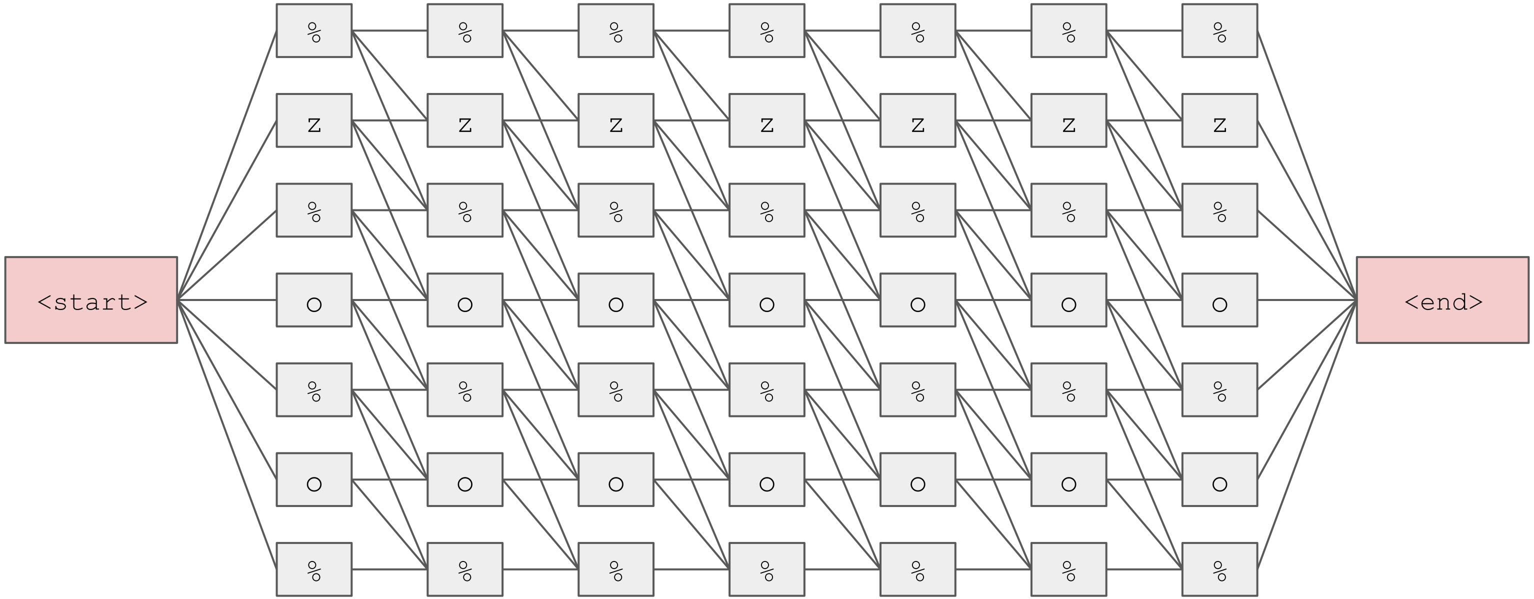
Probabilities

We want to transcribe "zoo" at these time steps

Target

“zoo”

(Training)



Not all of these are “viable” paths/alignments.
Which ones are?

“zoo”

(Training)

Let’s start from the outer edges and work our way in…

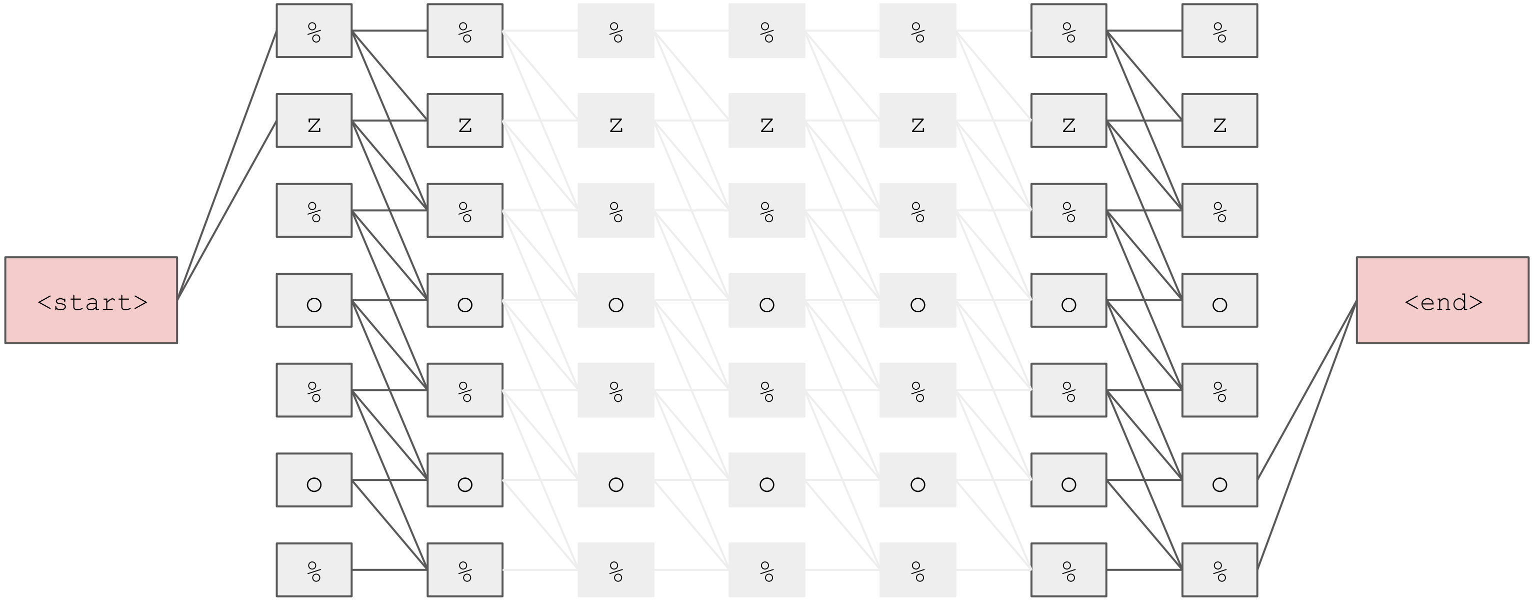
“zoo”

(Training)

Let’s start from the outer edges and work our way in…

"zoo"

(Training)

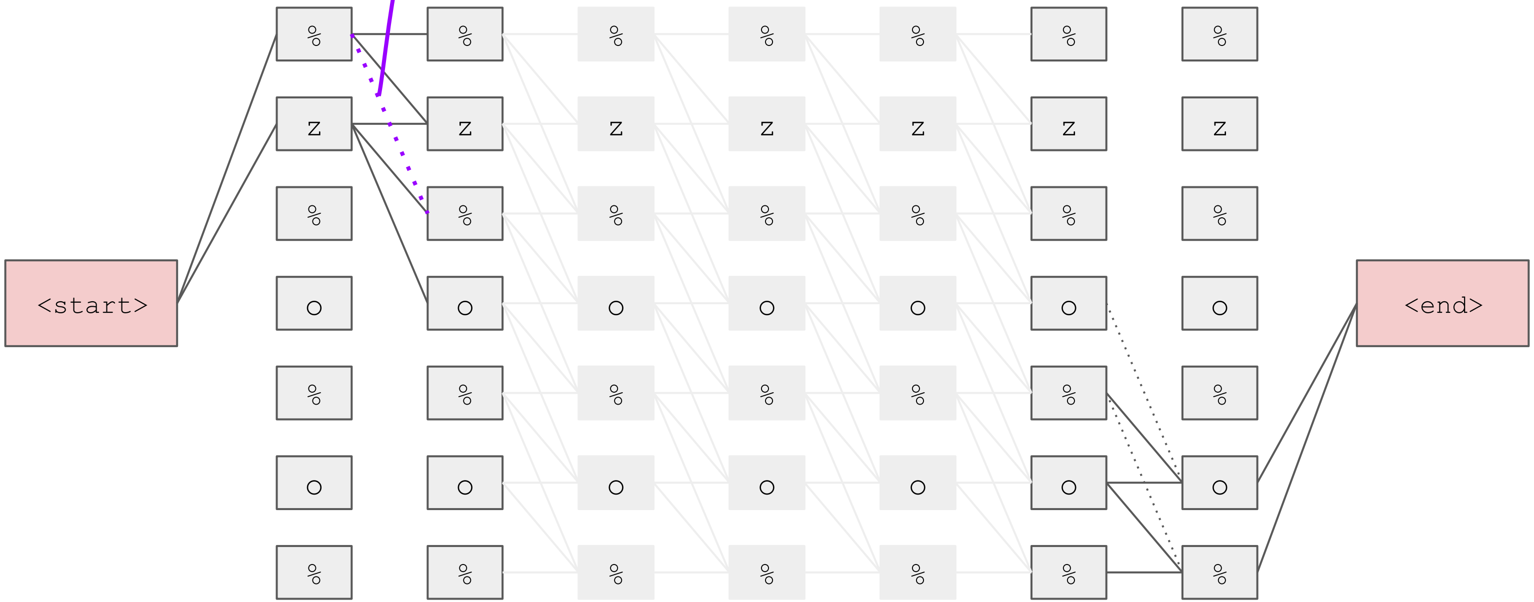


Let's start from the outer edges and work our way in...

“zoo”

(Training)

This path won’t work, because it would skip the “z” (not monotonic)

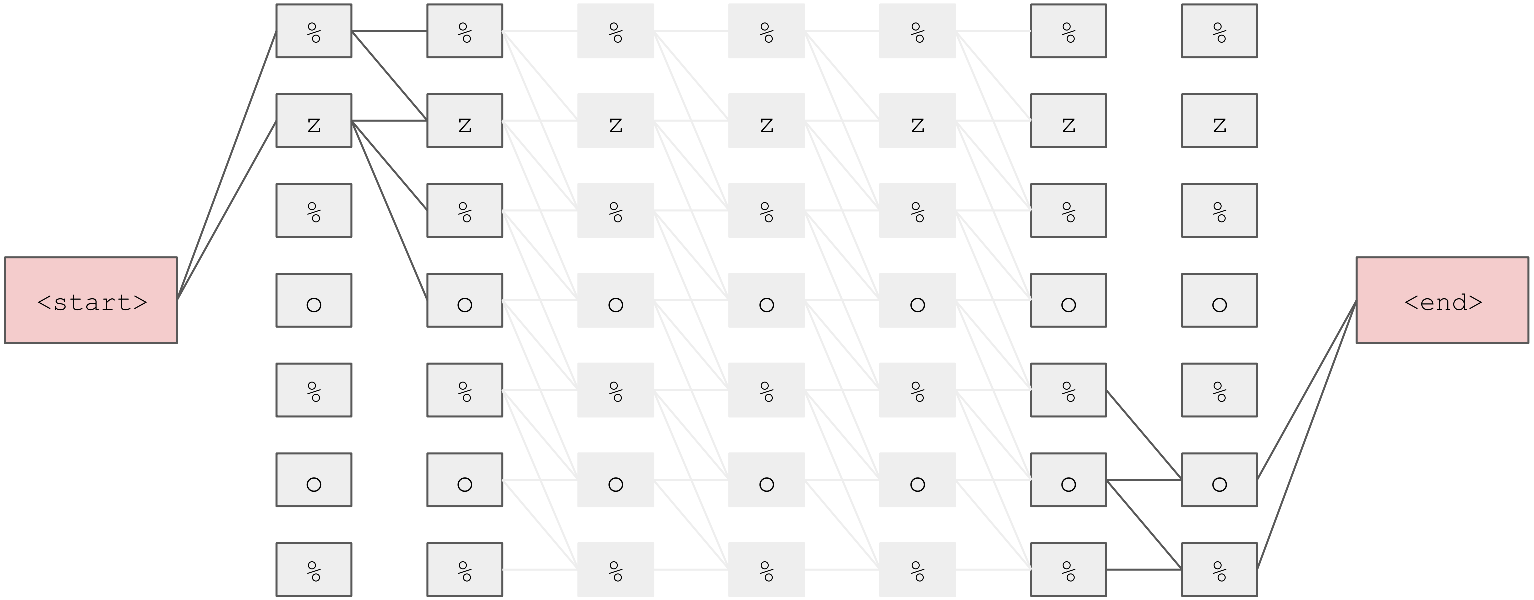


Dotted lines are paths that may seem viable at first, but are not!

Let’s start from the outer edges and work our way in...

“zoo”

(Training)



Let’s start from the outer edges and work our way in…

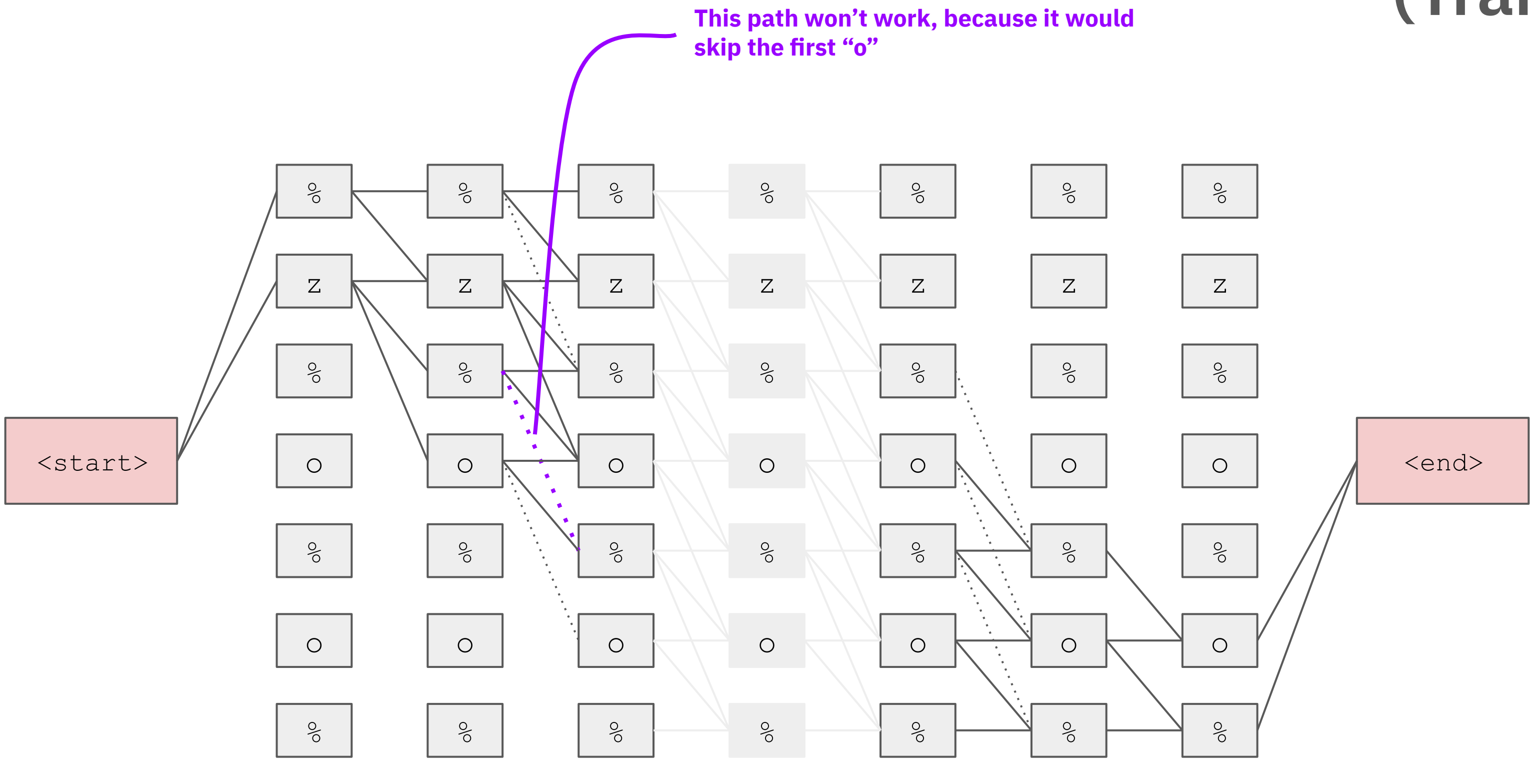
"zoo"

(Training)

Let's start from the outer edges and work our way in...

(Training)

“zoo”

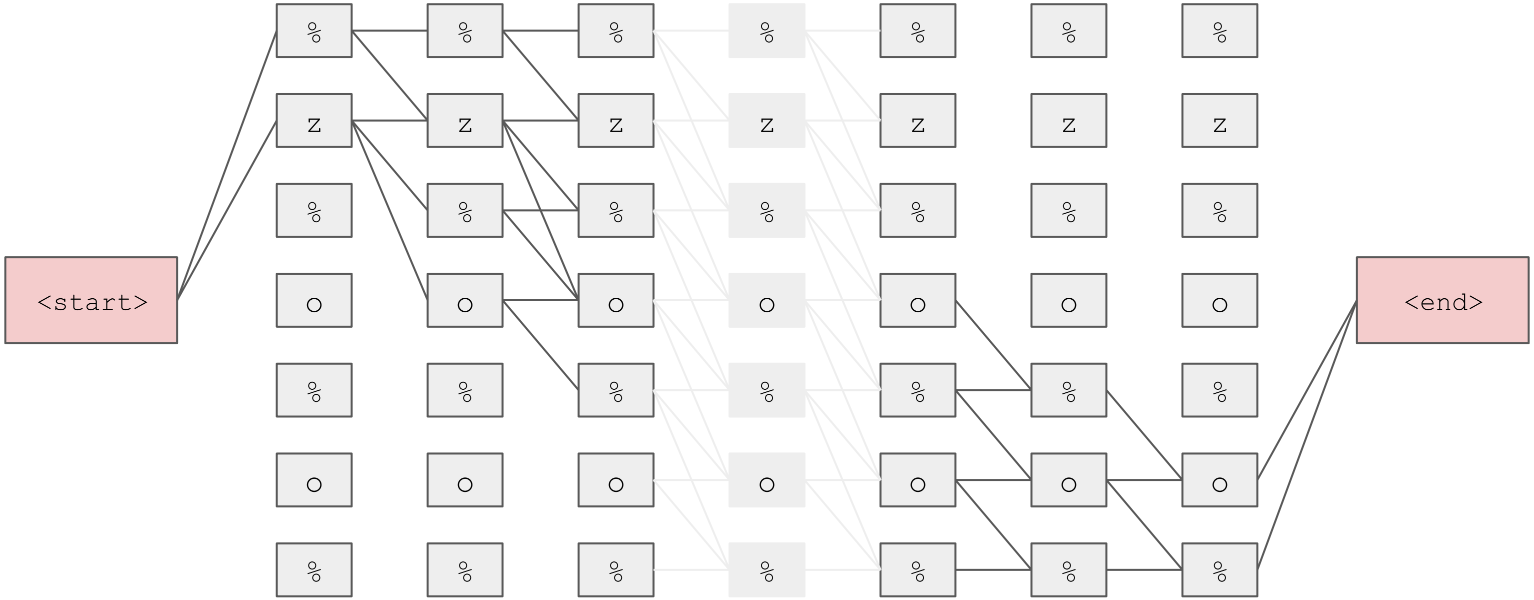


Dotted lines are paths that may seem viable at first, but are not!

Let’s start from the outer edges and work our way in...

"zoo"

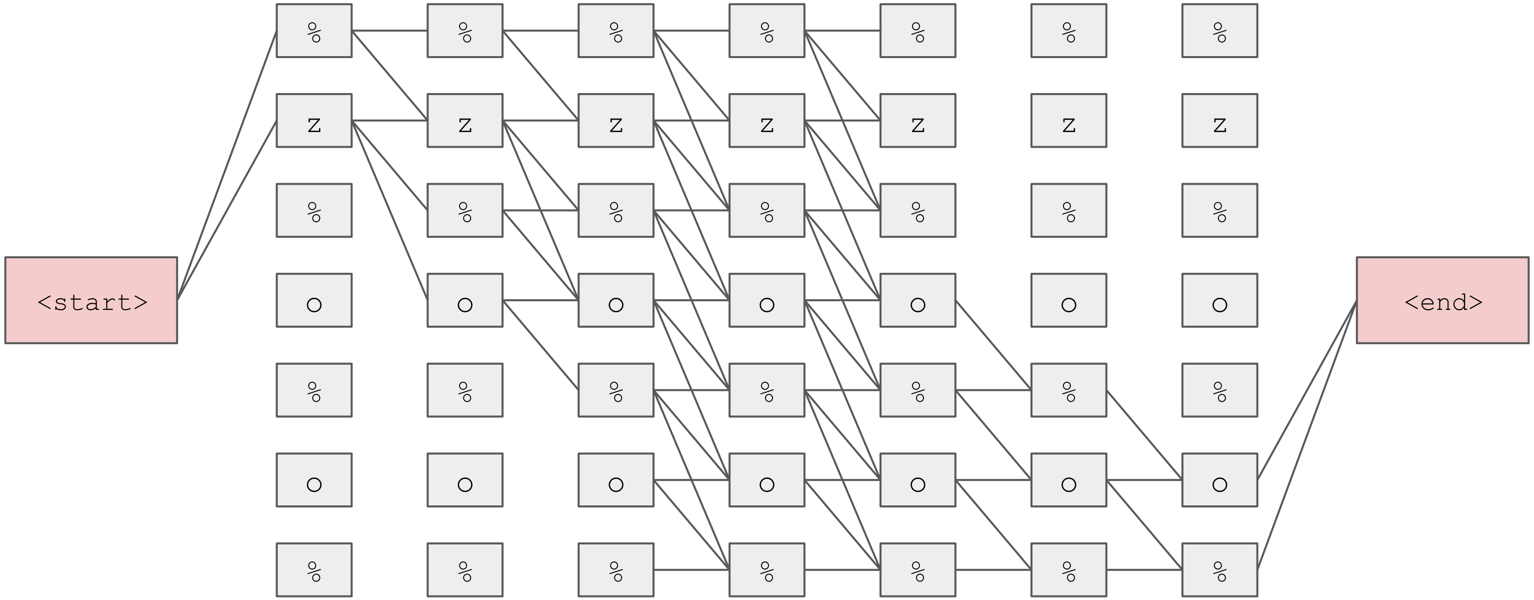
(Training)



Let's start from the outer edges and work our way in...

“zoo”

(Training)

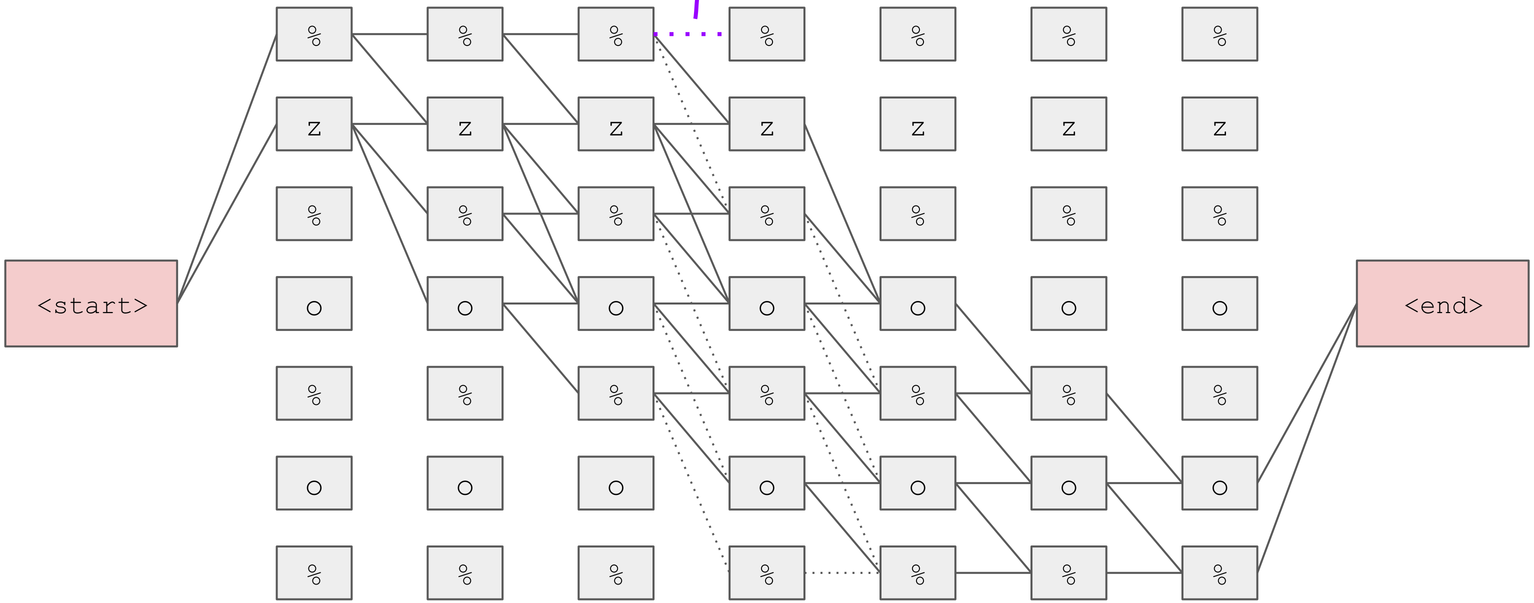


Let’s start from the outer edges and work our way in...

“zoo”

(Training)

This path won’t work because you won’t be able to get to the final character “in time” based on the sequence length.

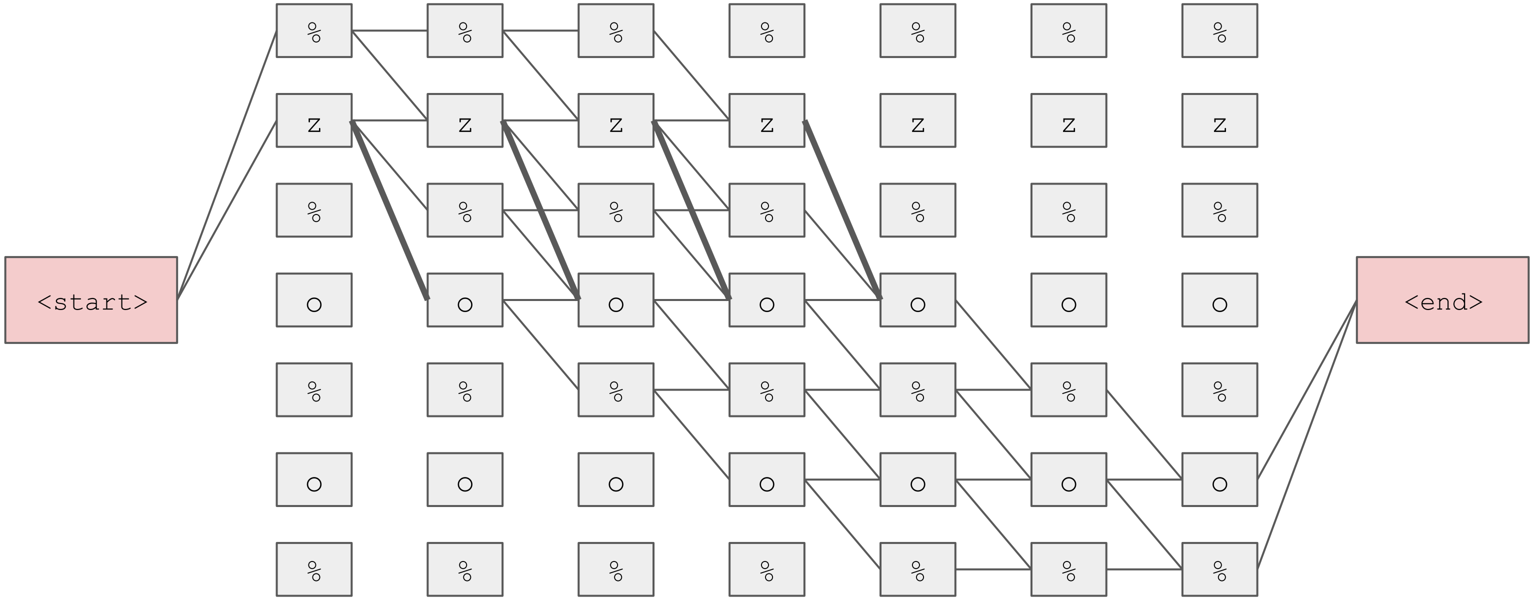


Dotted lines are paths that may seem viable at first, but are not!

Let’s start from the outer edges and work our way in...

"zoo"

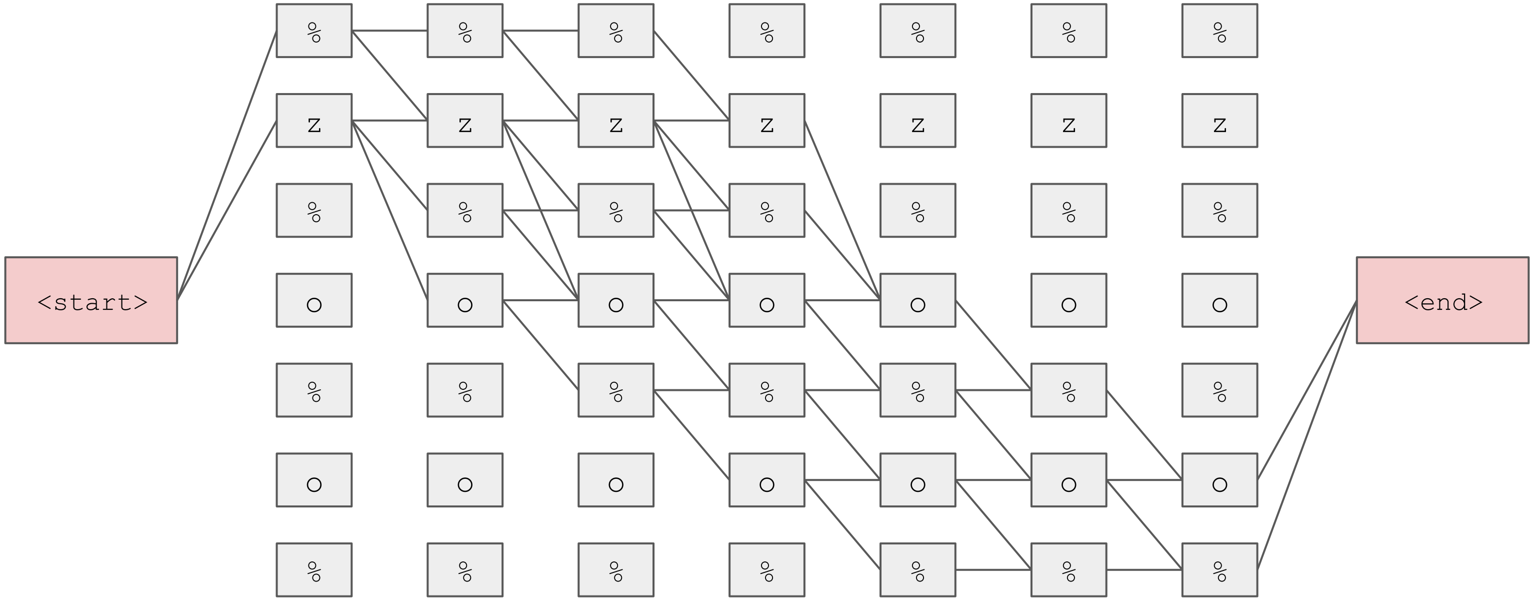
(Training)



These are the viable alignments
(skips bolded)

“zoo”

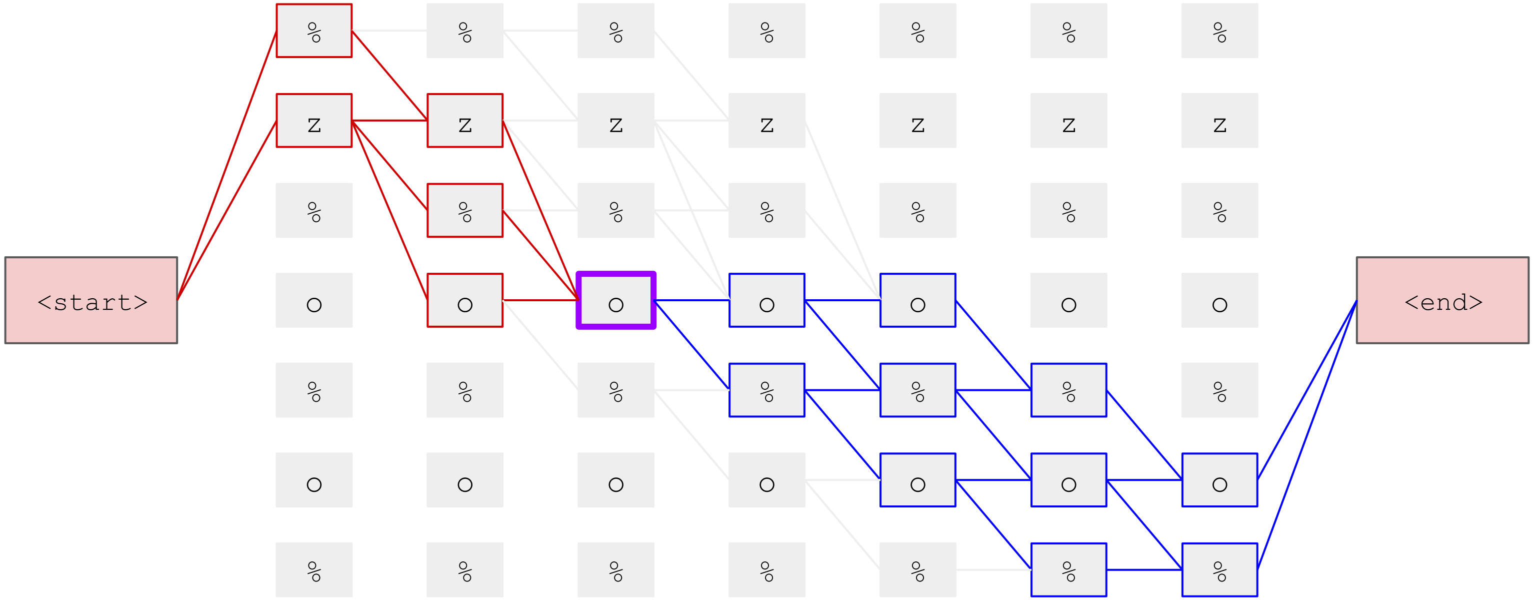
(Training)



These are the viable alignments

“zoo”

(Training)

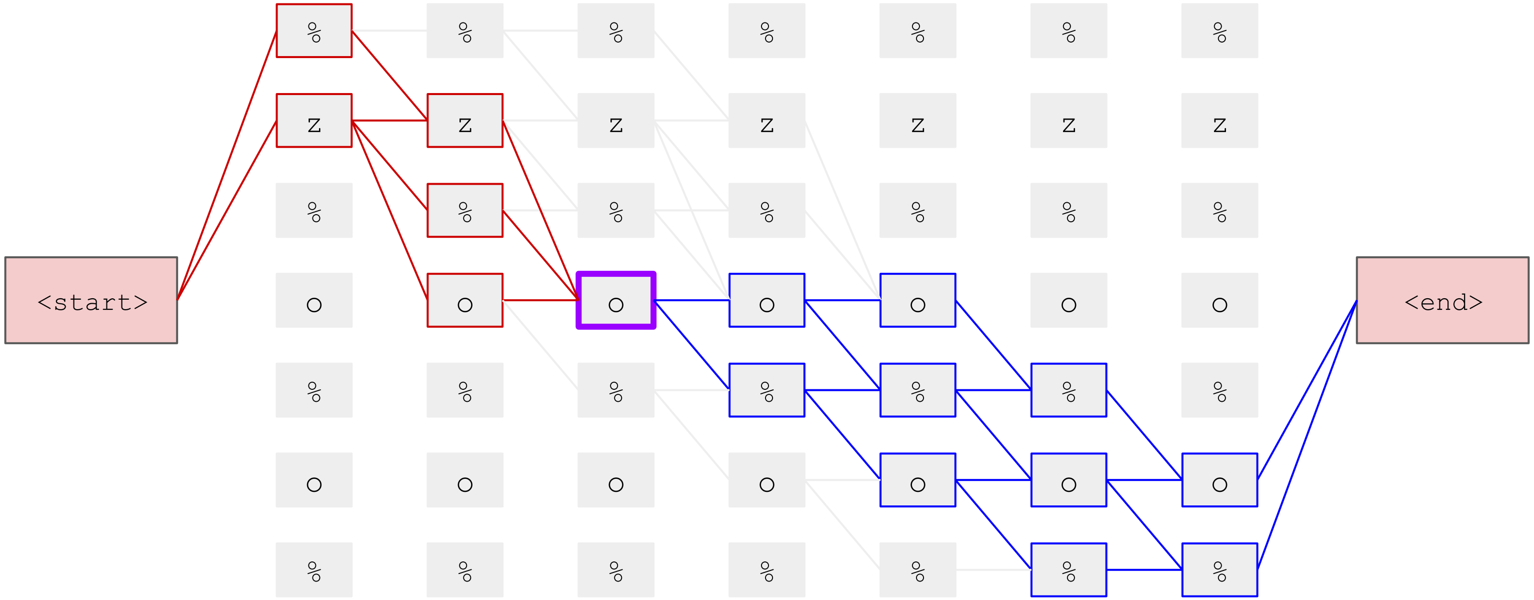


This is where dynamic programming comes in handy!

At any given node, you can calculate the posterior probability of reaching that node at that time step using the product of the probability of reaching from the “**forward**” and “**backward**” passes (See Lecture)

“zoo”

(Training)



We can calculate the probability of ALL ALIGNMENTS using the forward/backward method and use these to compute an EXPECTED LOSS.

“zoo”

(Training)

We can calculate the probability of ALL ALIGNMENTS using the forward/backward method and use these to compute an EXPECTED LOSS.

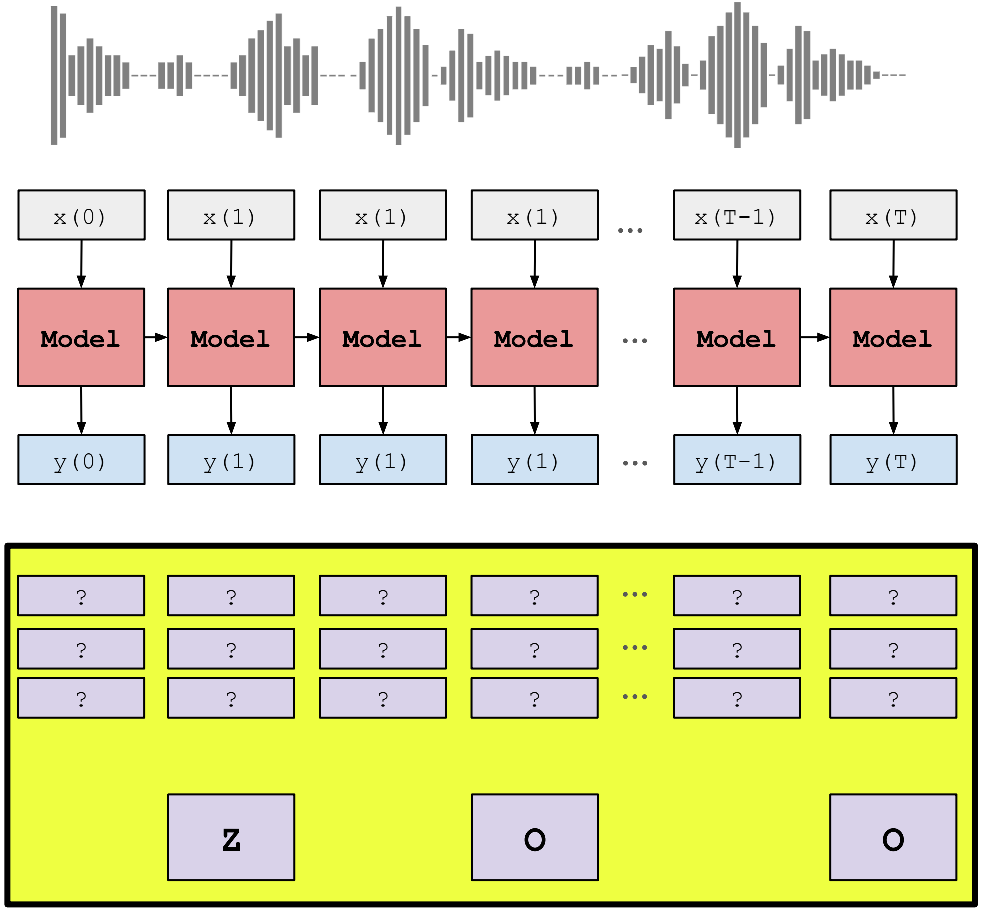
Training Procedure

With Connectionist Temporal Classification(CTC)

1. Define model (e.g., deep bidirectional LSTM)
2. Initialize network to output targets + “break” character
3. Pass training instances through network to obtain probability distribution over labels/symbols
4. Construct graph/table of “viable” alignments
5. Compute probabilities of alignments using Forward/Backward Algorithm (see Lecture)
6. Compute Expected Divergence over all alignments (see Lecture)
7. Propagate gradients backward and update parameters

(Training)

Someone says “zoo”



Inputs

Probabilities

Use **CTC** to get time-aligned targets

***All* time-aligned targets for Target**

We want to transcribe “zoo” at these time steps

Target

Inference

Q: What’s different at inference?

Q: What’s different at inference?

A: No target

A: No sequence “rules”

Q: What’s different at inference?

A: No target

A: No sequence “rules”

The “tree” is about to get very large :/ (gulp)

What are our options?

- Greedy Search
- Exhaustive Search (not really possible)
- Beam Search

Inference

Greedy Search

- Greedy Search is an easy-to-implement option for CTC decoding at inference time
- Greedy Search simply selects the most probable output at each time-step
- Although this method is easy to implement and fast, it has the disadvantage of missing out on high-probability (score) overall paths due to it's greedy search

Greedy Search

Y_probs

	T = 0	T = 1	T = 2	T = 3
-	0.140	0.257	0.248	0.149
A	0.391	0.096	0.402	0.336
B	0.197	0.341	0.267	0.358
C	0.271	0.305	0.083	0.157

DECODED STRING

?

Greedy Search

Y_probs

	T = 0	T = 1
-	0.140	0.257
A	**0.391**	0.096
B	0.197	**0.341**
C	0.271	0.305
	A	B

Greedy Search

Y_probs

	T = 0
-	0.140
A	**0.391**
B	0.197
C	0.271

A

Greedy Search

Y_probs

	T = 0	T = 1	T = 2
-	0.140	0.257	0.248
A	**0.391**	0.096	**0.402**
B	0.197	**0.341**	0.267
C	0.271	0.305	0.083
	A	B	A

Greedy Search

Y_probs

	T = 0	T = 1	T = 2	T = 3
-	0.140	0.257	0.248	0.149
A	**0.391**	0.096	**0.402**	0.336
B	0.197	**0.341**	0.267	**0.358**
C	0.271	0.305	0.083	0.157
	A	B	A	B

Greedy Search

Y_probs

	T = 0	T = 1	T = 2	T = 3
-	0.140	0.257	0.248	0.149
A	0.391	0.096	0.402	0.336
B	0.197	0.341	0.267	0.358
C	0.271	0.305	0.083	0.157

DECODED STRING

A B A B

0.391*0.341*0.402*0.358 =
0.0191884642

Beam Search

Greedy Search isn't perfect

target sequence: **the cat sat**

Vocab	**T=0**
the	0.3
cat	0.1
sat	0.1
hi	0.5

With these probabilities at t=0, greedy search **cannot** decode the correct sequence

How do we fix this ?

target sequence: **the cat sat**

Vocab	**T=0**
the	0.3
cat	0.1
sat	0.1
hi	0.5

Exhaustive search: Consider the **entire vocab** at each time step. Do not drop any paths.

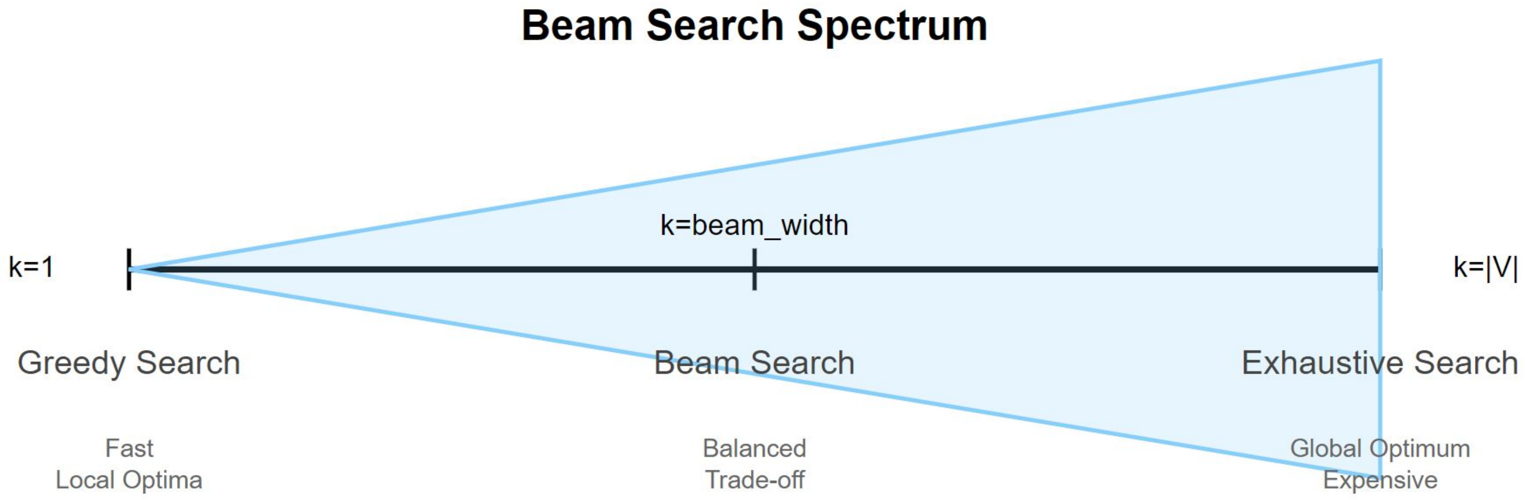
Beam search: a tradeoff (cost vs performance)

target sequence: **the cat sat**

Vocab	**T=0**
the	0.3
cat	0.1
sat	0.1
hi	0.5

Beam search: Take the **top k** at each time step. Drop other paths.

Beam Search: a tradeoff (cost vs performance)



Beam Search

Y_probs

	T = 0	T = 1	T = 2	T = 3
-	0.140	0.257	0.248	0.149
A	0.391	0.096	0.402	0.336
B	0.197	0.341	0.267	0.358
C	0.271	0.305	0.083	0.157

DECODED STRING

?

Parameters

Seq Len	4
Symbol set	{'-', 'A', 'B', 'C'}
Beam Width	3

Beam Search

BEAM WIDTH = 3

	T = 0
-	0.140
A	0.391
B	0.197
C	0.271

Scores

Possible Paths	Calculate Score	Score
-	0.140	0.140
A	0.391	0.391
B	0.197	0.197
C	0.271	0.271

Old

Best Paths	Score
–	–
–	–
–	–

Beam Search

BEAM WIDTH = 3

	T = 0
-	0.140
A	0.391
B	0.197
C	0.271

Scores

Possible Paths	Calculate Score	Score
-	0.140	0.140
A	0.391	0.391
B	0.197	0.197
C	0.271	0.271

Old

Best Paths	Score
–	–
–	–
–	–

New

Best Paths	Score
A	0.391
B	0.197
C	0.271

Beam Search

BEAM WIDTH = 3

	T = 1
-	0.257
A	0.096
B	0.341
C	0.305

Scores

Possible Paths	Calculate Score	Score
A-	0.391*0.257	0.10048700
B-	0.197*0.257	0.05062900
C-	0.271*0.257	0.06964700
AA -> A	0.391*0.096	0.03753600
AB	0.391*0.341	0.13333100
AC	0.391*0.305	0.11925500
BA	0.197*0.096	0.01891200
BB -> B	0.197*0.341	0.06717700
BC	0.197*0.305	0.06008500
CA	0.271*0.096	0.02601600
CB	0.271*0.341	0.09241100
CC -> C	0.271*0.305	0.08265500

Old

Best Paths	Score
A	0.391
B	0.197
C	0.271

Beam Search

BEAM WIDTH = 3

	T = 1
-	0.257
A	0.096
B	0.341
C	0.305

Scores

Possible Paths	Calculate Score	Score
A-	0.391*0.257	0.10048700
B-	0.197*0.257	0.05062900
C-	0.271*0.257	0.06964700
AA -> A	0.391*0.096	0.03753600
AB	0.391*0.341	0.13333100
AC	0.391*0.305	0.11925500
BA	0.197*0.096	0.01891200
BB -> B	0.197*0.341	0.06717700
BC	0.197*0.305	0.06008500
CA	0.271*0.096	0.02601600
CB	0.271*0.341	0.09241100
CC -> C	0.271*0.305	0.08265500

Old

Best Paths	Score
A	0.391
B	0.197
C	0.271

New

Best Paths	Score
A-	0.10048700
AB	0.13333100
AC	0.11925500

Beam Search

BEAM WIDTH = 3

	T = 2
-	0.248
A	0.402
B	0.267
C	0.083

Scores

Possible Paths	Calculate Score	Score
A–-	0.100487*0.248	0.0249207760
AB-	0.133331*0.248	0.0330660880
AC-	0.119255*0.248	0.0295752400
A-A -> AA	0.100487*0.402	0.0403957740
ABA	0.133331*0.402	0.0535990620
ACA	0.119255*0.402	0.0479405100
A-B -> AB	0.100487*0.267	0.0268300290
ABB -> AB	0.133331*0.267	0.0355993770
ACB	0.119255*0.267	0.0318410850
A-C -> AC	0.100487*0.083	0.0083404210
ABC	0.133331*0.083	0.0110664730
ACC -> AC	0.119255*0.083	0.0098981650

Old

Best Paths	Score
A-	0.10048700
AB	0.13333100
AC	0.11925500

Beam Search

BEAM WIDTH = 3

	T = 2
-	0.248
A	0.402
B	0.267
C	0.083

Scores

Possible Paths	Calculate Score	Score
A--	0.100487*0.248	0.0249207760
AB-	0.133331*0.248	0.0330660880
AC-	0.119255*0.248	0.0295752400
A-A -> AA	0.100487*0.402	0.0403957740
ABA	0.133331*0.402	0.0535990620
ACA	0.119255*0.402	0.0479405100
A-B -> AB	0.100487*0.267	0.0268300290
ABB -> AB	0.133331*0.267	0.0355993770
ACB	0.119255*0.267	0.0318410850
A-C -> AC	0.100487*0.083	0.0083404210
ABC	0.133331*0.083	0.0110664730
ACC -> AC	0.119255*0.083	0.0098981650

Old

Best Paths	Score
A-	0.10048700
AB	0.13333100
AC	0.11925500

New

Best Paths	Score
ABA	0.0535990620
ACA	0.0479405100
AB	0.0624294060

Beam Search

BEAM WIDTH = 3

	T = 3
-	0.149
A	0.336
B	0.358
C	0.157

Scores

Possible Paths	Calculate Score	Score
ABA- -> ABA	0.0535990620*0.149	0.0079862602380
ACA- -> ACA	0.0479405100*0.149	0.0071431359900
AB- -> AB	0.0624294060*0.149	0.0093019814940
ABAA -> ABA	0.0535990620*0.336	0.0180092848320
ACAA -> ACA	0.0479405100*0.336	0.0161080113600
ABA	0.0624294060*0.336	0.0209762804160
ABAB	0.0535990620*0.358	0.0191884641960
ACAB	0.0479405100*0.358	0.0171627025800
ABB -> AB	0.0624294060*0.358	0.0223497273480
ABAC	0.0535990620*0.157	0.0084150527340
ACAC	0.0479405100*0.157	0.0075266600700
ABC	0.0624294060*0.157	0.0098014167420

Old

Best Paths	Score
ABA	0.0535990620
ACA	0.0479405100
AB	0.0624294060

Beam Search

BEAM WIDTH = 3

	T = 3
-	0.149
A	0.336
B	0.358
C	0.157

Scores

Possible Paths	Calculate Score	Score
ABA- -> ABA	0.0535990620*0.149	0.0079862602380
ACA- -> ACA	0.0479405100*0.149	0.0071431359900
AB- -> AB	0.0624294060*0.149	0.0093019814940
ABAA -> ABA	0.0535990620*0.336	0.0180092848320
ACAA -> ACA	0.0479405100*0.336	0.0161080113600
ABA	0.0624294060*0.336	0.0209762804160
ABAB	0.0535990620*0.358	0.0191884641960
ACAB	0.0479405100*0.358	0.0171627025800
ABB -> AB	0.0624294060*0.358	0.0223497273480
ABAC	0.0535990620*0.157	0.0084150527340
ACAC	0.0479405100*0.157	0.0075266600700
ABC	0.0624294060*0.157	0.0098014167420

Old

Best Paths	Score
ABA	0.0535990620
ACA	0.0479405100
AB	0.0624294060

New

Best Paths	Score
ABA	0.04697182548
AB	0.031651708842
ACA	0.02325114735

Beam Search

Remember, when extending a path with a new symbol, you'll encounter three scenarios:

1. The new symbol is the same as the last symbol on the path.
2. The last symbol of the path is blank.
3. The last symbol of the path is different from the new symbol and is not blank.

Efficient Beam Search:

```
Input: SymbolSets, y_probs, BeamWidth
Output: BestPath, MergedPathScores

  0. Initialize:
     1. BestPaths with a blank symbol path with a score of 1.0.
     2. TempBestPaths as an empty dictionary.
  1. For each timestep in y_probs:
     1. Extract the current symbol probabilities.
     2. For each path, score in BestPaths limited by BeamWidth:
        1. For each new symbol in the current symbol probabilities:
           1. Based on the last symbol of the path, determine the new path.
           2. Update the score for the new path in TempBestPaths.

     3. Update BestPaths with TempBestPaths.
     4. Clear TempBestPaths.

  2. Initialize MergedPathScores as an empty dictionary.

  3. For each path, score in BestPaths:
     1. Remove the ending blank symbol from the path.
     2. Update the score for the translated path in MergedPathScores.
     3. Update the BestPath and BestScore if the score is better.

  4. Return BestPath and MergedPathScores.
```

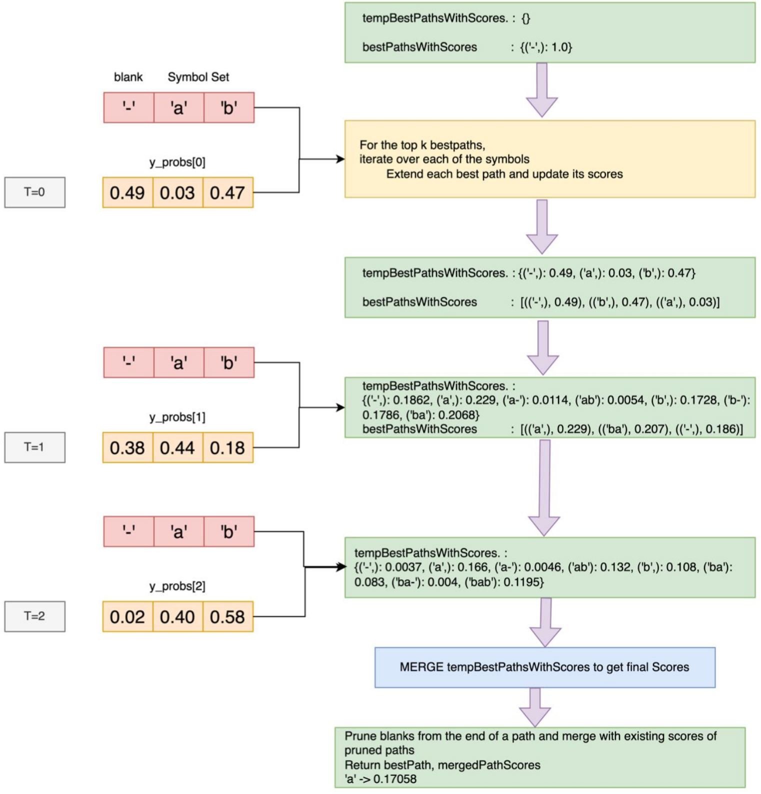
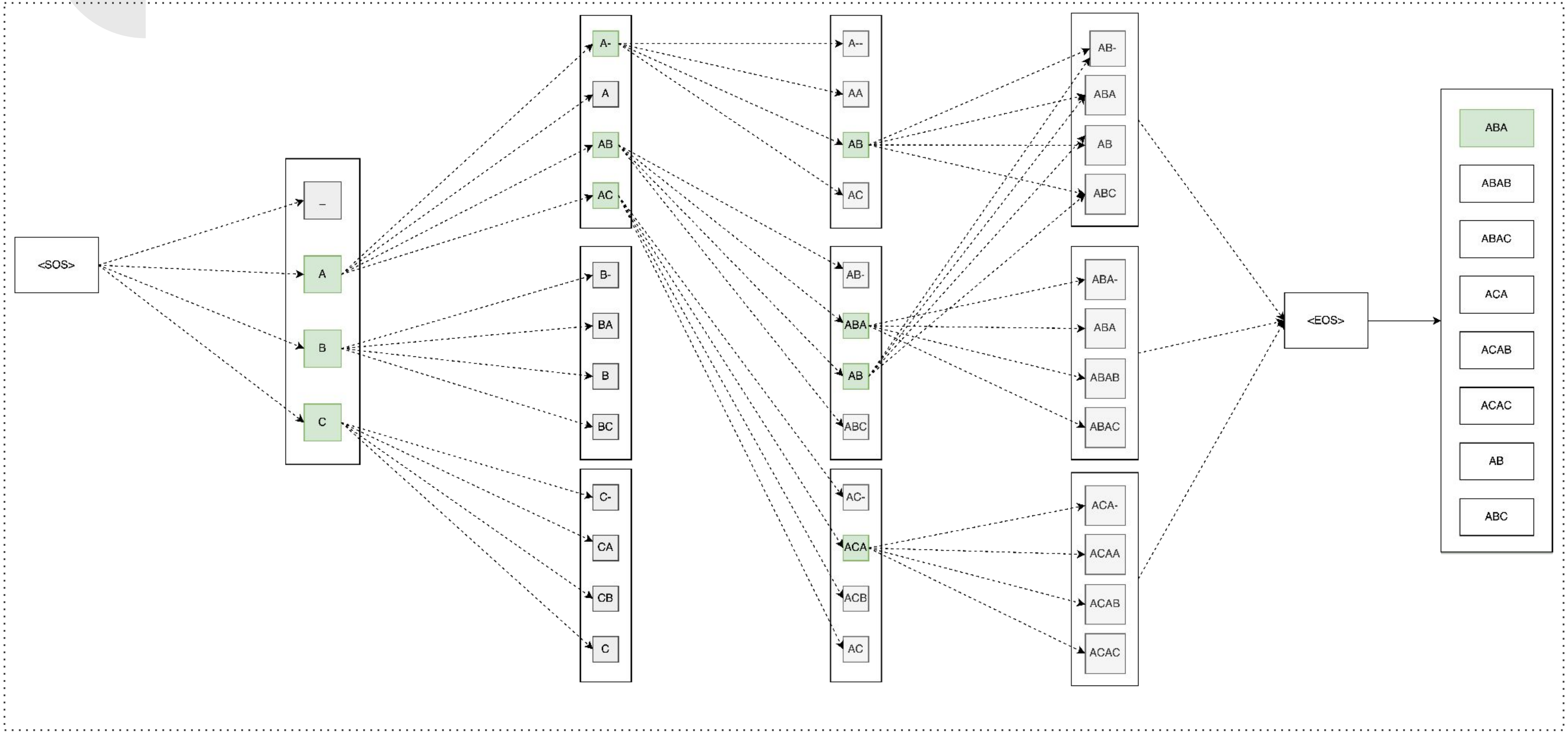


Figure 20: Efficient Beam Search procedure

Beam Search

BEAM WIDTH = 3



Extra notes

- Beam search (or even exhaustive search) **isn’t an alternative** to sufficiently training a model to output good probabilities.
- Because of the additional time complexity of considering more than 1 path, it is common to use a **smaller beam width** (or even greedy search) **for validation** and a **larger beam width for inference**.

Something to think about…

Can we expect monotonic performance improvement with k?

If not, why?

See:

1. Beam Search: Faster and Monotonic
2. Empirical Analysis of Beam Search Performance Degradation in Neural Sequence Models
3. Breaking the Beam Search Curse