SCHOOL OF DATA ANALYSIS

ASR III: Discriminative State-Space Models II

Andrey Malinin

21th March 2022

Story so far

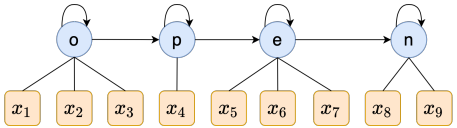
In this previous episode...

Recap - Data Processing

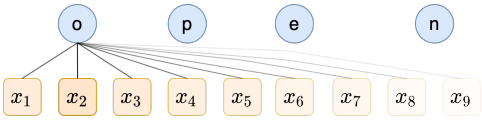
- Transform audio into Mel-Spectrogram
  - Yields sequence of acoustic features or frames $\boldsymbol{X}_{1:T}$
  - This decreases the time-resolution (downsamples) of the audio data
  - Length of each frame depends on audio sample-rate and hop-length
- Transform text into a sequence of speech units $\omega_{1:L}$
  - Speech units can be phonemes, gramphes, syllables, etc..
  - Speech units must be compact and easy to back to / from words

Problem of Alignment

- Which feature vectors $\boldsymbol{X}_{1:T}$ and speech units $\boldsymbol{\omega}_{1:L}$ relate or align to each other?
  - Models need to be able to dynamically align features vectors to words.
- Two common approaches to constructing models which can align
  - State-Space models
  - **Neural Attention Mechanisms**

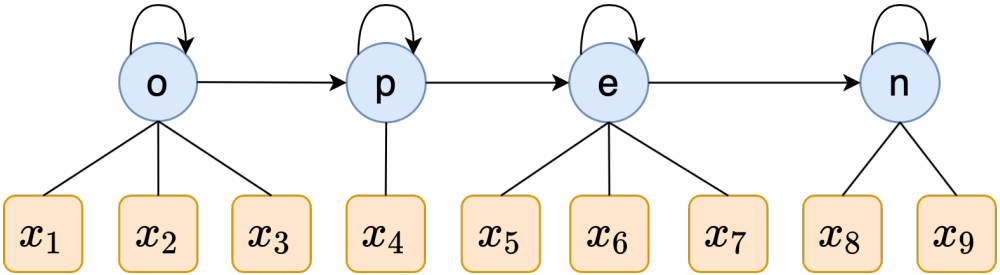


(a) State-Space



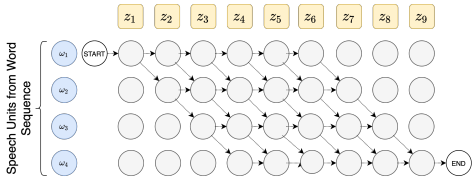
(b) Attention Mechanism

State-Space Alignment Models

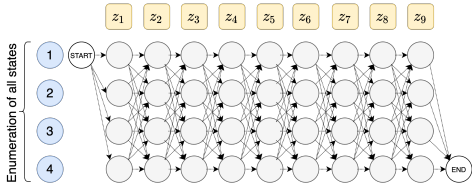


State-Space Trellis Structure

- Training / alignment and inference and trellises have different structure!
  - Training / Alignment Trellis - speech unit sequence defines set of valid paths
  - Inference Trellis - Any path is valid



(c) Training / Alignment



(d) Inference

Discriminative State-Space ASR Models

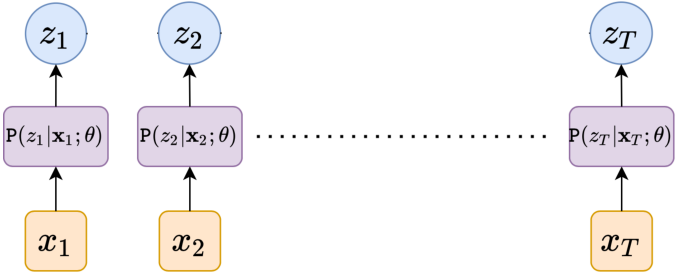
- **Discriminative** ASR $\rightarrow$ "Which sequence $\hat{\boldsymbol{w}}$ is likely given the audio?"

$$\hat{\boldsymbol{w}} = \mathcal{M}^{-1}(\hat{\omega}_{1:L}), \quad \hat{\omega}_{1:L} = \arg\max_{\hat{\omega}_{1:L}} \mathrm{P}(\hat{\omega}_{1:L} | \boldsymbol{X}_{1:T}; \boldsymbol{\theta})$$

- **Generative** ASR $\rightarrow$ "Given which sequence $\hat{\boldsymbol{w}}$ is the audio mostly likely?"

$$\hat{\boldsymbol{w}} = \mathcal{M}^{-1}(\hat{\omega}_{1:L}), \quad \hat{\omega}_{1:L} = \arg\max_{\hat{\omega}_{1:L}} \mathrm{P}(\boldsymbol{X}_{\boldsymbol{1:T}} | \hat{\omega}_{1:L}; \boldsymbol{\theta}) \mathrm{P}(\hat{\omega}_{1:L}; \boldsymbol{\theta})$$

Connectionist Temporal Classification (CTC)



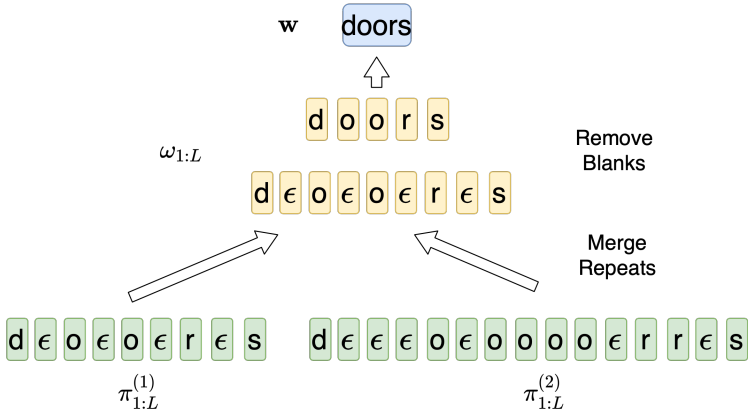
- CTC is a discriminative state-space ASR model defined as:

$$\mathrm{p}(\boldsymbol{w}|\boldsymbol{X}_{1:T};\boldsymbol{\theta}) = \sum_{\boldsymbol{\pi}_{1:T} \in \mathcal{A}(\boldsymbol{\omega}_{1:L}, T)} \prod_{t=1}^{T} \mathrm{p}(z_t|\boldsymbol{x}_t;\boldsymbol{\theta}),\ \boldsymbol{\omega}_{1:L} = \mathcal{M}(\boldsymbol{w})$$

Connectionist Temporal Classification (CTC) - Properties

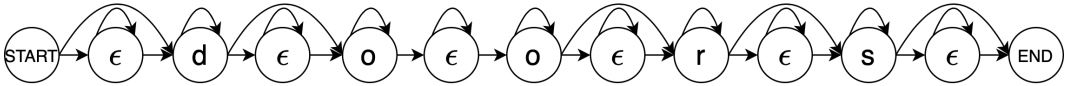
- CTC assumes all states are conditionally independent
  - Can have poor context modelling
- Alignment free - Do not need prior alignment for training
  - Train by maximizing probability of all valid alignments
- CTC models are typically graphemic with an extra black symbol ϵ.
  - Allows simple mapping between words and speech units

Connectionist Temporal Classification (CTC)



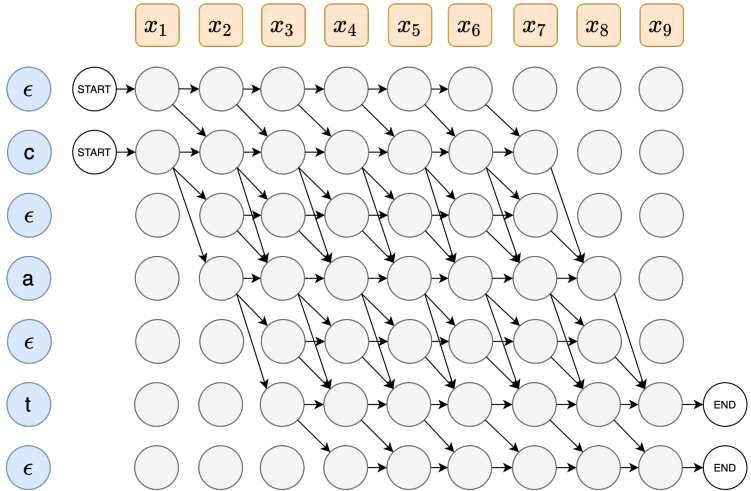
- Introduce a **blank** symbol $\epsilon \rightarrow$ allows for repeats
  - Restore works by first merging repeats, then removing blanks.

Connectionist Temporal Classification (CTC)



- For training use a sequence $\omega'_{1:2L+1}$ with a blank before and after each character
  - Sequence $\omega'_{1:2L+1}$ contains all valid alignments.
- Following state transitions in are $\omega'_{1:2L+1}$ valid:
  - Self-transitions
  - Transitions between blank and non-blank
  - Transitions between distinct non-blank labels

Connectionist Temporal Classification (CTC) - Alignment and Training Trellis



Limitations of CTC

Greedy Decoding

- CTC doesn't model inter-state dependencies $\rightarrow$ independently take arg-maxes:

$$\mathrm{P}(\boldsymbol{z}_{1:T}|\boldsymbol{X}_{1:T}) = \prod_{t=1}^{T} \mathrm{P}(z_t|\boldsymbol{X}_{1:T})$$

$$\boldsymbol{\pi}^*_{1:T} = \arg\max_{\boldsymbol{\pi}_{1:T}} \prod_{t=1}^{T} \mathrm{P}(z_t = \pi_t|\boldsymbol{X}_{1:T})$$

- This yields most probable state sequence - then merge repeats and remove blanks.
- GD can still fail to find the most likely $\boldsymbol{\omega}^*_{1:L}$
  - Most likely path $\boldsymbol{\omega}^*_{1:L}$ may be composed of multiple, less likely state sequences.
  - Grammatical constraints not enforced!

Example of CTC Output

there se ms no good reason for believing that twillc ange — Hypothesis

there seems no good reason for believing that it will change — Reference

- Clearly, the output is almost correct
  - Letters 'sound' ok is pronounced, but orthography is wrong and space misplaced

Example of CTC Output

the dinamble alectoric machi in thoh small was robused four under all the varyin speeds of waterdpower and the tdessicitudes of the plant to which hat belonged it continued in activuse until eighteen nighty nine seventy nears

Hypothesis

the dynamo electric machine though small was robust for under all the varying speeds of water power and the vicissitudes of the plant to which it belonged it continued in active use until eighteen ninety nine seventeen years

Reference

CTC has issues with context modelling

Example of CTC Output

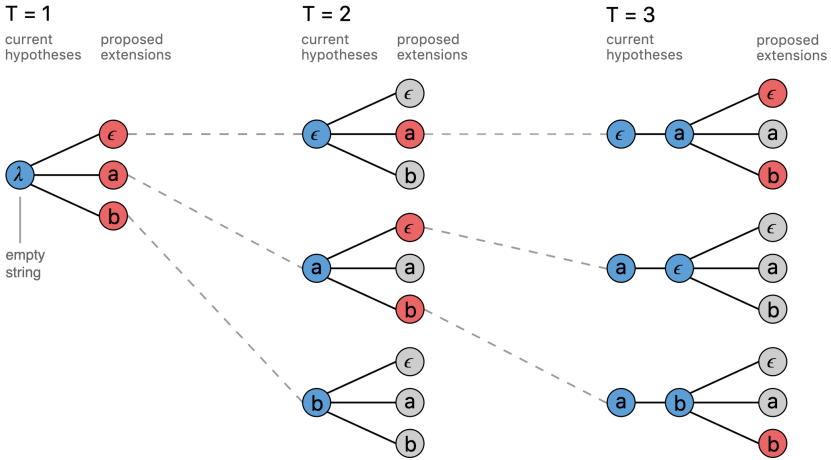
- CTC does not explicitly model output-output interactions
  - Output is 'phonetically' close, but orthographically poor
  - How can we improve this and enforce grammatical and orthographic constraints?
- Improve Inference - use **Prefix Beam-Search**
  - Can enforce orthographic constraints via **external vocabulary**
  - Can use an external **language model**
- Explicitly model output-output interactions via **RNN-Transducer**
  - Simpler trellis structure
  - Initialization from pre-trained Language Model
  - Output not constrained by number of frames.

Prefix Beam Search Decoding

Prefix Beam Search Decoding

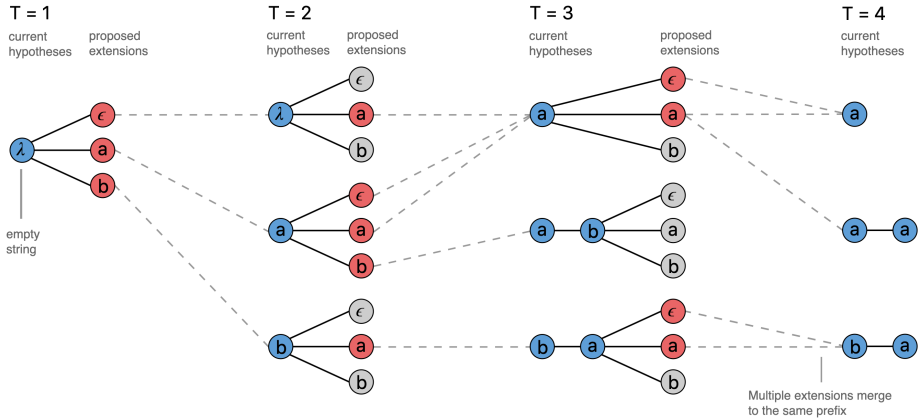
- In CTC, many paths $\pi_{1:T}$ map to the same character sequence $\omega_{1:L}$
  - Greedy Decoding finds only most likely path $\pi^*_{1:T}$
  - Most likely character sequence $\omega^*_{1:L}$ could be composed of many less-likely paths!
- Prefix decoding considers probabilities of multiple paths and merges them!
  - Maintain beam of B most likely prefixes
  - Extend beam, then merge probabilities which map into same prefix
  - Can add external language model

Prefix Beam Search Decoding



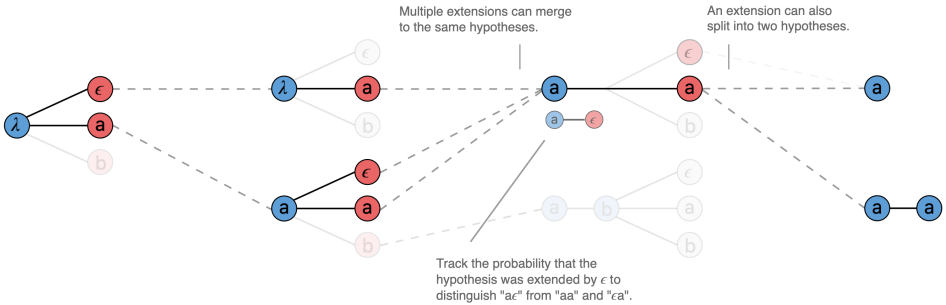
A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.

Prefix Beam Search Decoding



The CTC beam search algorithm with an output alphabet $\{\epsilon, a, b\}$ and a beam size of three.

Prefix Beam Search Decoding



- We are extending the prefix $\rightarrow$ could have ended with blank
- Must keep track of two probabilities per prefix:
  - Probability of prefix ending in blank $\mathtt{P_b}$
  - Probability of not ending in blank $\mathtt{P_{nb}}$

Prefix Beam Search Decoding - Overview

- Prefix Decoder has 3 iterations - over time, prefixes in beam and vocab
  1. Iterate over time - we extend the prefix up to T times
  2. Iterate over the prefixed in the beam
  3. Iterate over the vocab - can prune
- We iterating over vocab, we have 3 extension cases:
  1. Extend with blank
  2. Extend with repeat
  3. Extend with non-repeat (if space - add LM)
- For each case, we update the probabilities of ending with blank / non blank
- After iterating over prefixes we select B best prefixes for the next iteration
- After iterating over time, select best prefix

Prefix Beam Search Decoding - Step 1: Initialization

- We initialize iteration with empty string prefix
- Set $\texttt{Pb}_0(\texttt{1}) = 1$ and $\texttt{Pnb}_0(l) = 0$
  - Empty prefix could only happen if we came here from an 'imaginary blank'
- Start our 3 nested iterators
  - When iterating over vocab, can add a pruning step

Prefix Beam Search Decoding - Step 2: Extend with Blank

- If we extend with a blank - update the probability of ending with blank
  - Do not actually extend the prefix l - blanks get removed!
- Update probabilities as follows:

$$\mathrm{Pb}_t(l) = \mathrm{P}(\epsilon|\boldsymbol{x}_t)(\mathrm{Pb}_{t-1} + \mathrm{Pnb}_{t-1}(l)$$

Prefix Beam Search Decoding - Step 3: Extend with repeat character

- Consider extending with the last character C of the prefix - two cases
  1. Previously we had a blank, and now we extend the prefix
  2. Previously we had no blank, so we don't extend prefix (repeats are merged)
- Update probabilities as follows:

$$\mathrm{Pnb}_t(l + c) = \mathrm{P}(C|\boldsymbol{x}_t) \cdot \mathrm{Pb}_{t-1}(l)$$
$$\mathrm{Pnb}_t(l) = \mathrm{P}(C|\boldsymbol{x}_t) \cdot \mathrm{Pnb}_{t-1}(l)$$

Prefix Beam Search Decoding - Step 4: Extend with Non-Blank

- Consider extending prefix l at time t with a non-repeat character
  - Could have come from both blank and non-blank
- Update probabilities as follows:

$$\mathrm{Pnb}_t(l + c) = \mathrm{P}(C|\boldsymbol{x}_t) \cdot (\mathrm{Pb}_{t-1}(l) + \mathrm{Pnb}_{t-1}(l))$$

Prefix Beam Search Decoding - Step 5: Select Next Beam

- Select top B best prefixes based on combined score $\texttt{Pnb}_t(l) + \texttt{Pb}_t(l)$
- Restart iteration

Prefix Beam Search Decoding - Summary

- Prefixed Beam-Search can yield improvements in performance, when:
  - Model is not ‘sharp’ and small beams useful
  - When combined with LM or vocabulary
- However, can significantly slow down inference

5 minute break!

RNN-Transducer - Explicit context modelling

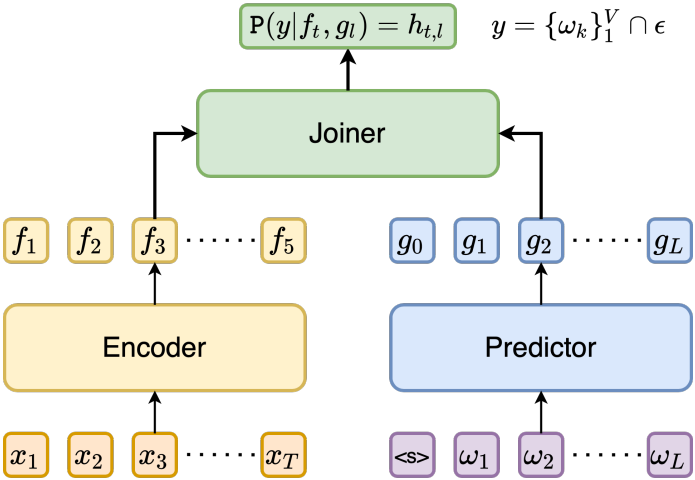
RNN Transducer - Architecture

- RNN Transducer is a discriminative state-space ASR model

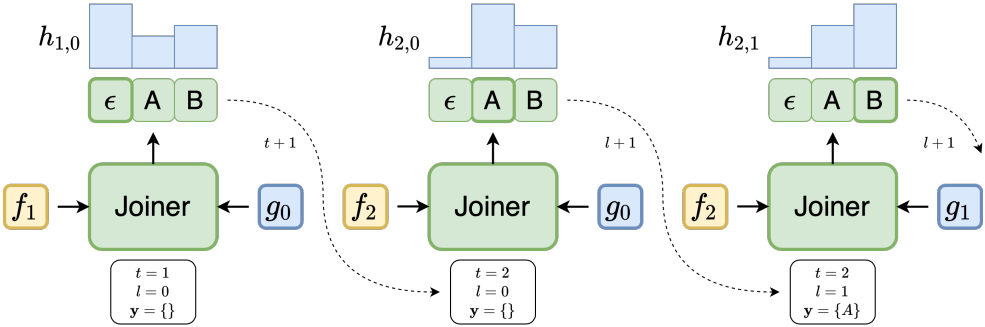
$$\begin{aligned} \mathrm{P}(\boldsymbol{\omega_{1:L}}|\boldsymbol{x}_{1:T}) &= \sum_{\boldsymbol{\pi}_{0:T+L} \in \mathcal{A}(\boldsymbol{\omega}_{1:L})} \mathrm{P}(\boldsymbol{\pi}_{0:T+L}|\boldsymbol{x}_{1:T}), \quad \boldsymbol{\omega}_{1:L} = \mathcal{M}(\boldsymbol{w}) \\ &= \sum_{\boldsymbol{\pi}_{0:T+L} \in \mathcal{A}(\boldsymbol{\omega}_{1:L})} \prod_{i=0}^{T+L} \mathrm{P}(\pi_i|\boldsymbol{x}_{1:T}), \qquad \pi \in \{\omega_k\}_{k=1}^{V} \cap \epsilon \end{aligned}$$

- It's primary differences from CTC are the following:
  - Explicitly models context and structure and feature level
  - Can **yield** sequences of arbitrary length and **align** sequences of length T+L
  - Simpler lattice structure
  - Initialization from pre-trained CTC and LM models
- It's main limitation is increased computational cost

RNN Transducer - Architecture

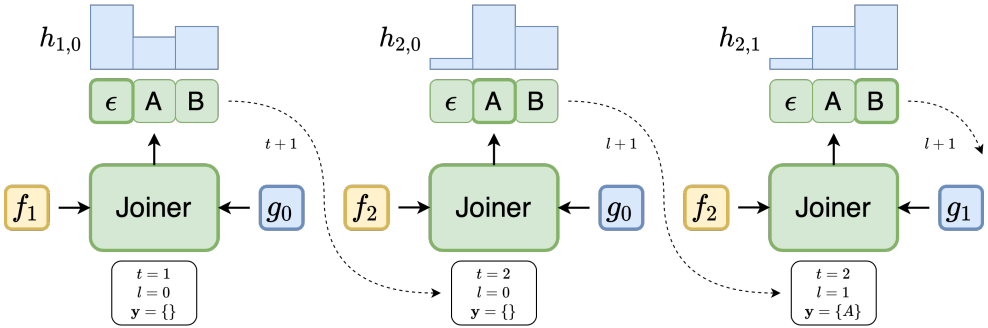


RNN Transducer Inference - Greedy Decoding



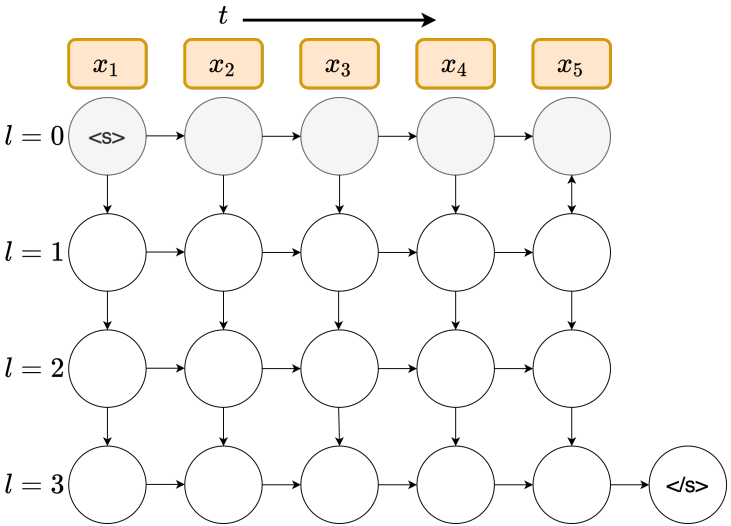
- Begin with empty prefix. Acoustic frame into $t = 1$, context index $l = 0$.
  - If ϵ is predicted $\rightarrow$ increment acoustic frame index t.
  - If character is predicted $\rightarrow$ increment l, append character to prefix.

RNN Transducer Inference - Greedy Decoding

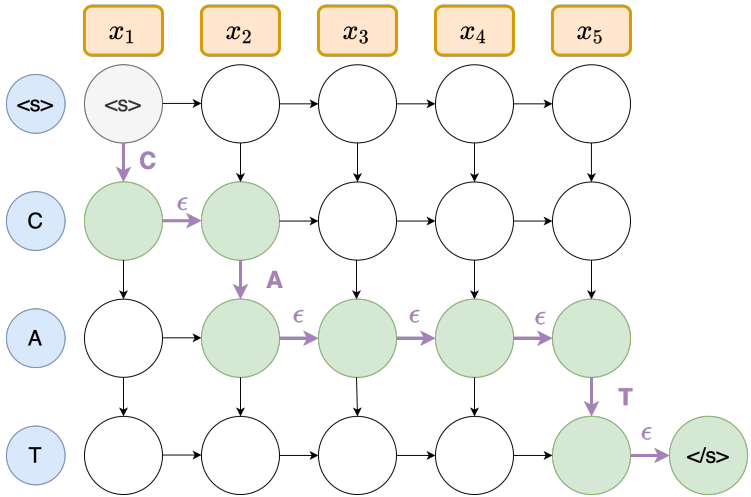


- Greedy path - not guaranteed to be most likely - beam-search can do better.
  - Beam-search can be expensive - requires re-running predictor network
- If encoder and predictor are non-bidirectional can do streaming inference
  - Good for stream-ASR applications, such as vocal assistants.

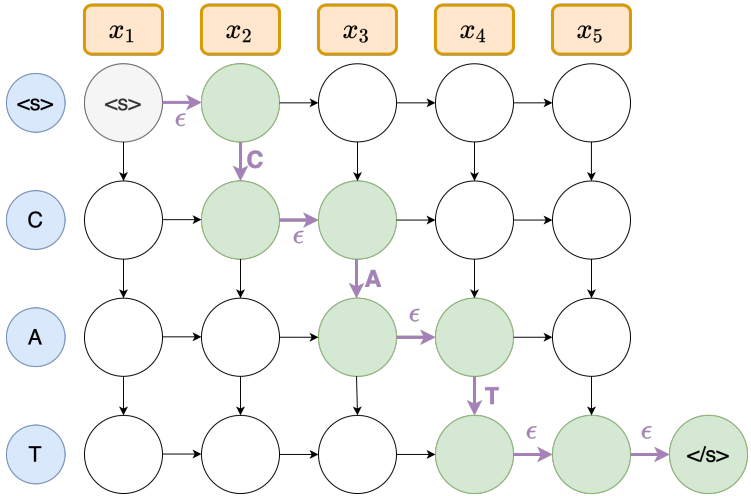
RNN Transducer Alignment Trellis



RNN Transducer Alignment Trellis



RNN Transducer Alignment Trellis



RNN Transducer - Loss-Computation and the Forward-Backward Algorithm

- Like CTC, RNN-T is **alignment free** $\rightarrow$
  - Don't need prior knowledge of how $\omega 1 : L$ aligns to $\boldsymbol{x}_{1:T}$
- RNN-T model is trained to maximize the probability of all valid alignments:

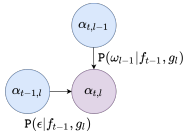
$$\begin{aligned}\mathcal{L} = & -\ln \mathrm{P}(\boldsymbol{\omega_{1:L}} | \boldsymbol{x}_{1:T}^{(n)}) \\ = & -\ln \sum_{\boldsymbol{\pi}_{0:T+L} \in \mathcal{A}(\boldsymbol{\omega}_{1:L})} \prod_{i=0}^{T+L} \mathrm{P}(\pi_i | \boldsymbol{x}_{1:T})\end{aligned}$$

- Naive summation is impractical - use dynamic programming
  - RNN-T Forward-Backward Algorithm

RNN Transducer - Loss-Computation and the Forward-Backward Algorithm

- Forward variable $\alpha_{t,l}$ computes the probability of generating $\omega_{0:l}$ using frames $\boldsymbol{x}_{1:t}$.

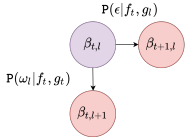
$$\alpha_{t,l} = \alpha_{t-1,l}\mathrm{P}(\epsilon|g_l, f_{t-1}) + \alpha_{t,l-1}\mathrm{P}(\omega_{l-1}|g_{l-1}, f_t)\alpha_{1,0} = \quad 1$$



(e) Forward Rule

- Backwards variable $\beta_{t,l}$ computes the probability of generating $\omega_{l+1:L}$ using frames $\boldsymbol{x}_{t:T}$.

$$\beta_{t,l} = \beta_{t+1,l}\mathrm{P}(\epsilon|g_l, f_t) + \beta_{t,l+1}\mathrm{P}(\omega_l|g_l, f_t)$$
$$\beta_{T,L} = \mathrm{P}(\epsilon|g_L, f_T)$$

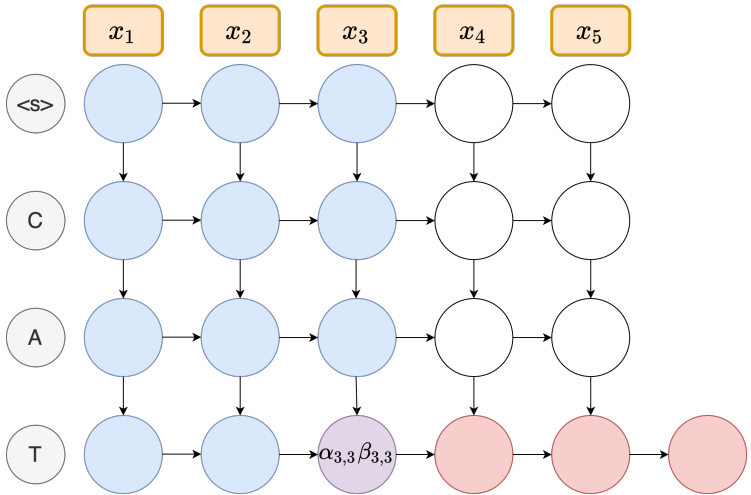


(f) Backward Rule

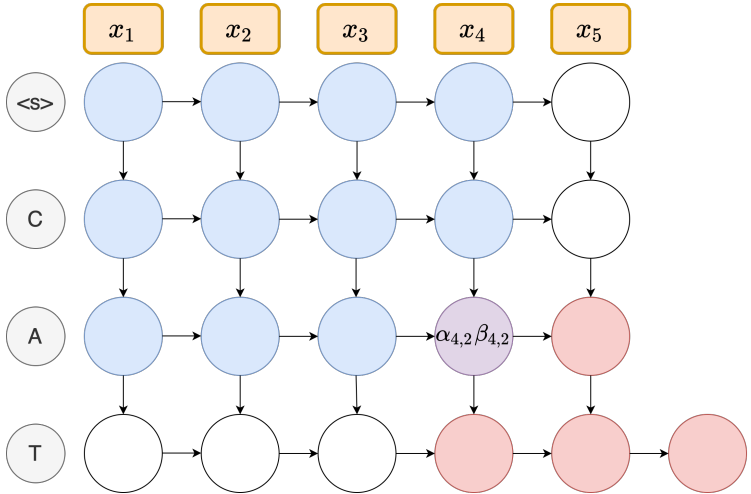
- Probability of sequence $\boldsymbol{omega}_{1:L}$ is given by:

$$\mathrm{P}(\boldsymbol{\omega_{1:L}}|\boldsymbol{x}_{1:T}^{(n)}) = \sum_{n=t+l):N=T+L} \alpha_{t,l}\beta_{t,l}$$

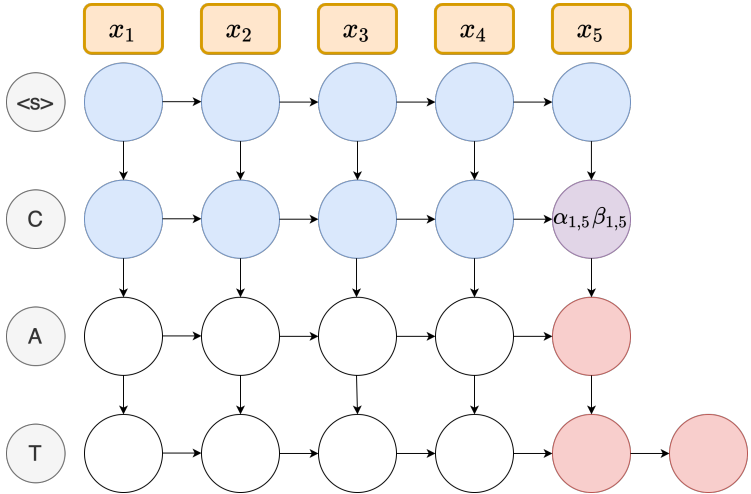
RNN Transducer - Forward-Backward Algorithm



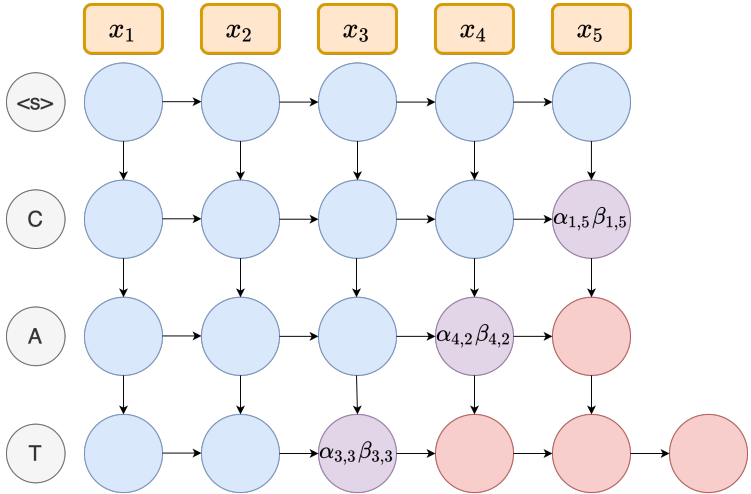
RNN Transducer - Forward-Backward Algorithm



RNN Transducer - Forward-Backward Algorithm



RNN Transducer - Forward-Backward Algorithm



RNN Transducer - Forward-Backward Algorithm

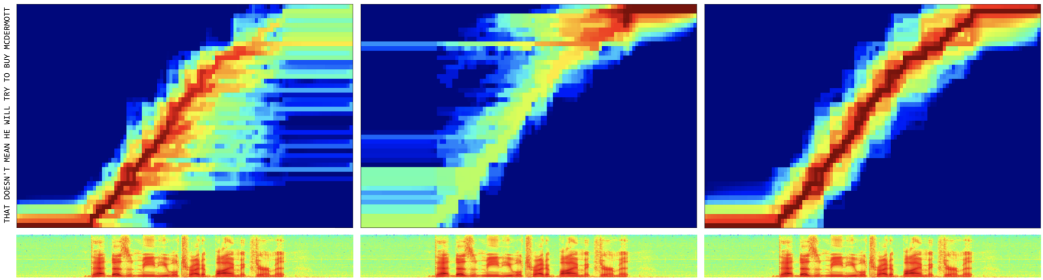


Figure from `https://arxiv.org/pdf/1211.3711.pdf`

Conclusion

- CTC has poor context modelling and makes orthographic mistakes
  - Can improve inference (Prefix Beam Search) or model (RNN-T)
- Prefix Beam Search - improved inference procedure
  - Allows finding globally likely sequences $\omega_{1:L}$
  - Allows adding vocab restrictions or using LMs
  - Can be slow
- RNN-Transducer
  - Implicitly models context
  - Can generate arbitrarily long sequences
  - Can be pre-trained
  - Good for streaming ASR