RNN-T Based ASR Systems

Mahaveer Jain, Facebook

Outline

- Automatic Speech Recognition (ASR) Systems
- Different Approaches For Building ASR Systems
- RNN-T (Recurrent Neural Network Transducer) Based ASR Training And Decoding
- Practical Considerations For deploying RNN-T ASR Systems in Production

Use pptx version if possible to see slides in Slide Show (Animation View) for best viewing as there are animations throughout

Automatic Speech Recognition

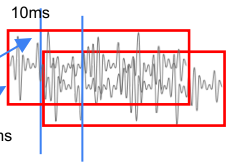
features → the cat sat

Audio

Transcript

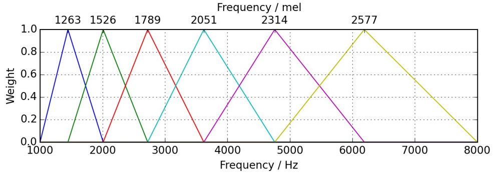
Audio → Overlapping Audio Frames → Frequency Domain Representation

Fourier Transform



Frame Shift

Frame Window



This filter bank image is taken from pyFilterBank documentation page http://siggigue.github.io/

Mel Filters : Humans are better at detecting differences in lower frequencies than higher frequencies

Mel Features → Log

→ X_1 X_2 X_3 X_4 X_5 X_6 X_7 …… X_T

Audio

Audio Feature at frame 3

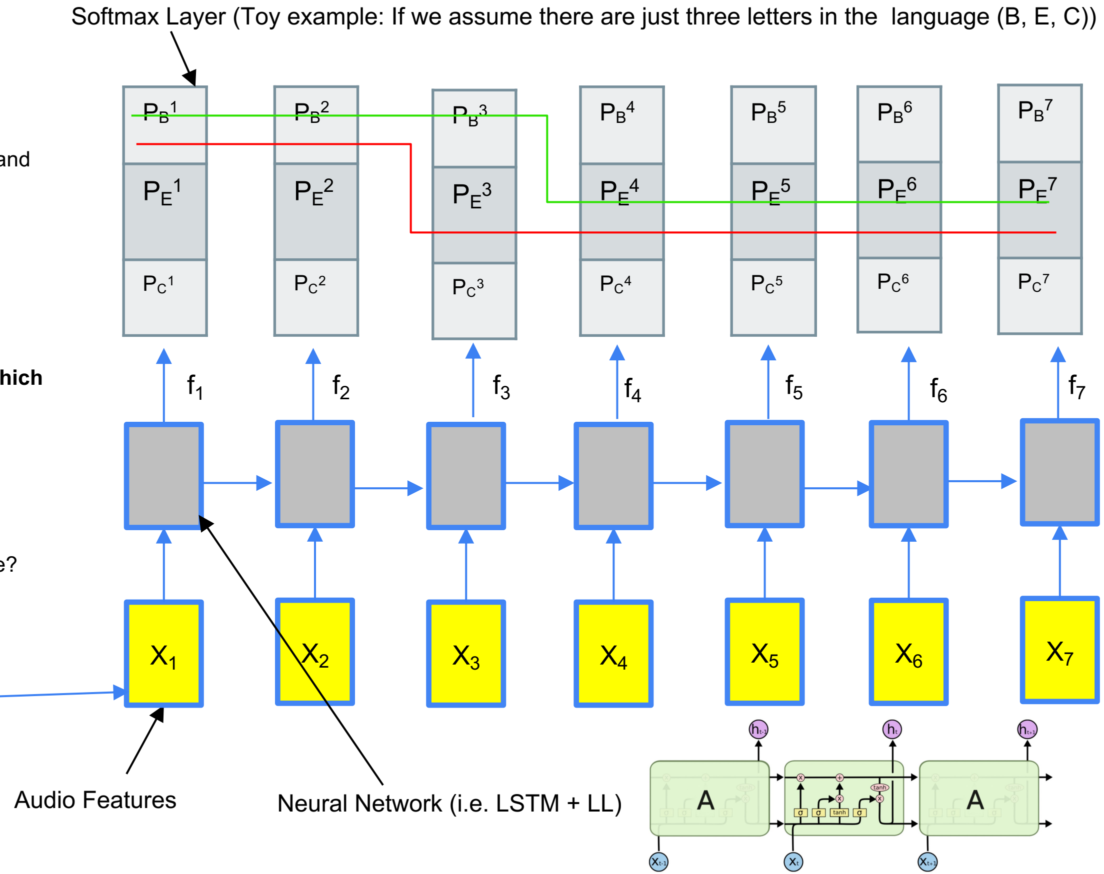
LSTM image is taken from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

ASR As Sequence To Sequence Problem

- Input Sequence: Audio frame features
- Output Sequence : Words from True Transcript Text written as letters
- Example Training Instance: Input Sequence : 7 audio frames and True Transcript Text: “BE”
- At each audio frame index, network* generates probability of producing each output symbol. E. g. Probability of generating output symbol “K” at 1st frame.

$$P_K^1 = prob(s_1 = K \mid f_1)$$

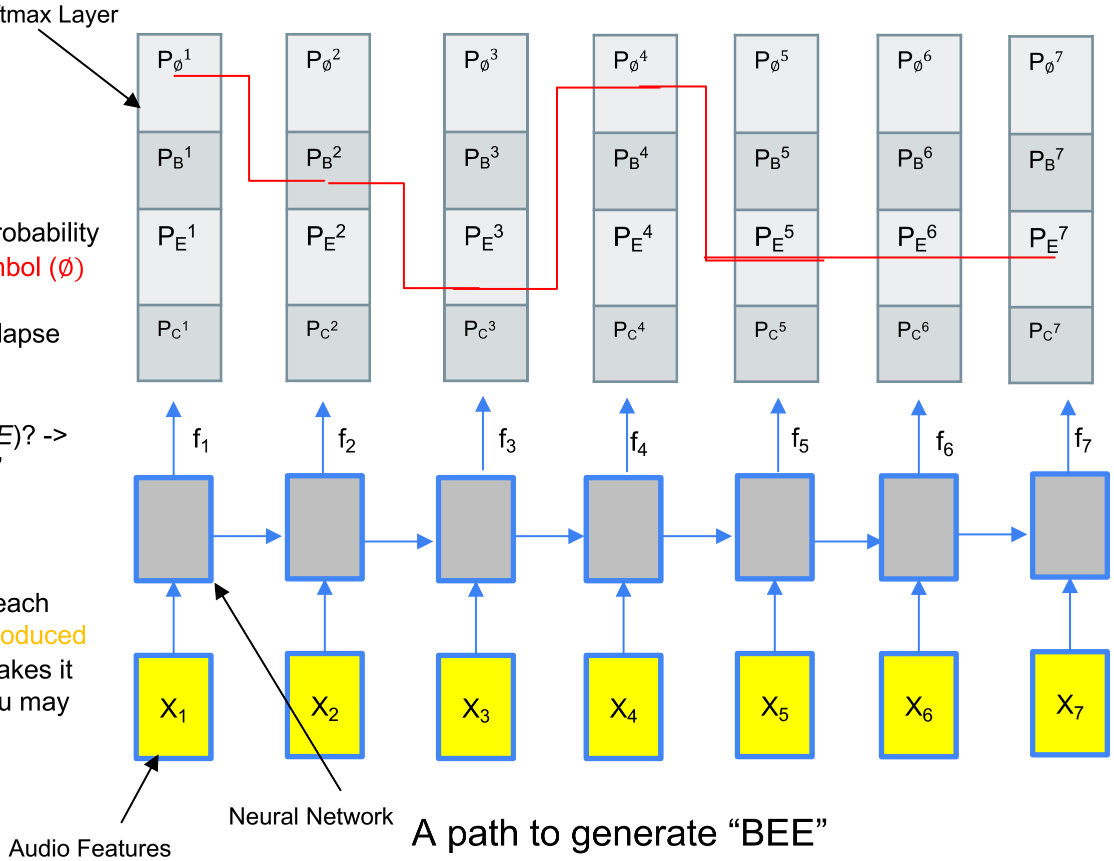
- **#audio frames != # letters. We don’t know alignment i.e. which portion of audio aligns to what letter in true transcript.**
- Collapse continuous occurrences of same letter into one:
  - First alignment choice (BBEEEEEE -> BE)
  - Second alignment choice (BBBEEEE -> BE)
  - and so on…..
- How to deal with repeated letters (as in word *BEE*) and slience? (next slide)



network* -> Softmax layer and neural network that produces f_t

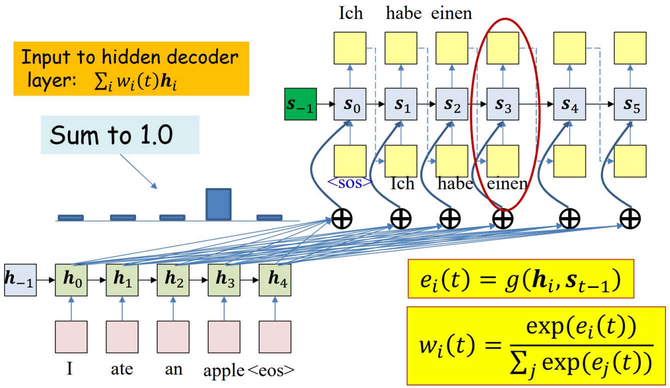
Connectionist Temporal Classification (CTC)

- Input Sequence: Audio frame features
- Output Sequence: Letters
- At each audio frame index, network generates probability of producing each output symbol and blank symbol (Ø)
- How to deal with #audio frame != # letters -> Collapse letters
- How to deal with repeated letters (as in word *BEE*)? -> blank symbol (Ø) . Ø also indicates "emit nothing" (silence). ØBEØEEØ
- Conditional Independence: Probability output at each frame does not depend on history of transcript produced so far, this non auto-regression on text history makes it less effective on learning LM information and, you may still need language model -> RNN-T.



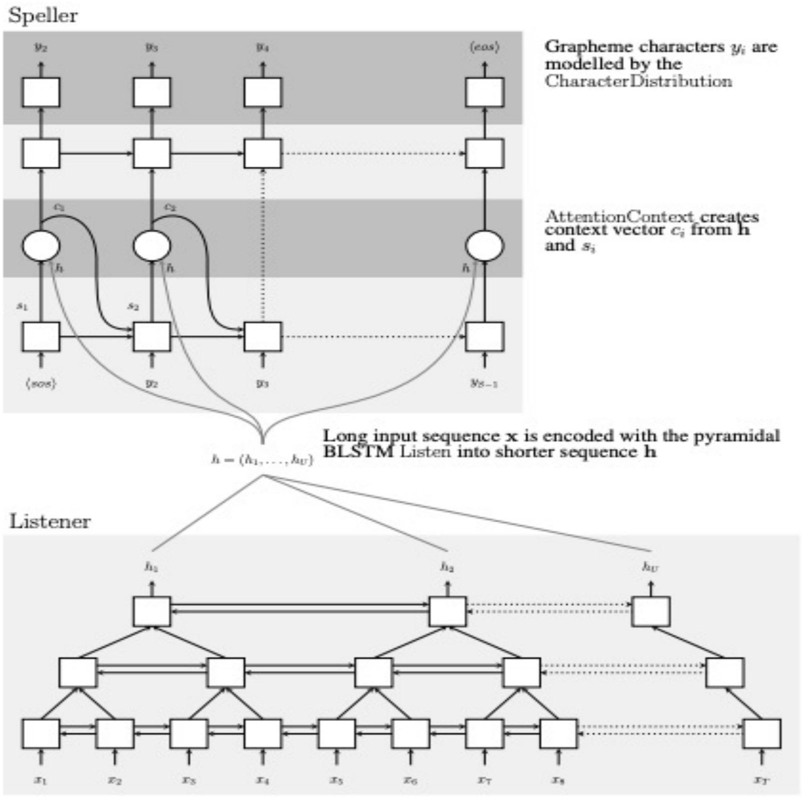
A path to generate "BEE"

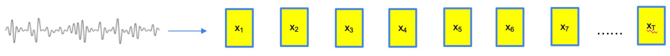
ASR And Machine Translation As Sequence To Sequence Problem



Machine Translation

Image from Bhiksha Raj's lecture on attention models





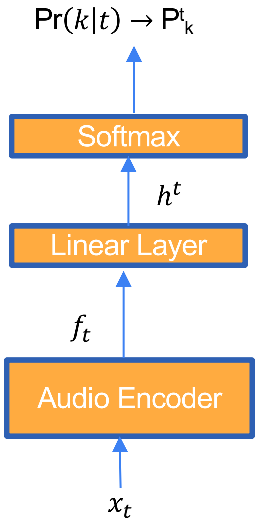
ASR

Image from Chan et al., "Listen, Attend and Spell"

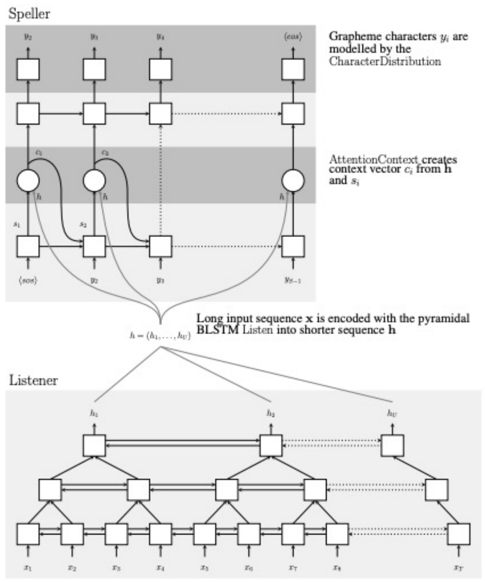
LAS attends to all audio embeddings and uses text history produced so far to generate probability distribution over output units

$$f(y_u | y_{u-1}, y_{u-2}, \ldots, y_0, h_0, h_2, h_3, \ldots, h_v)$$

Conditional Independence Assumption



>



CTC:

- Only uses audio embedding at time "t" (x_t) to generate probability distribution over output units. CTC assumes that each output unit is conditionally independent of others.

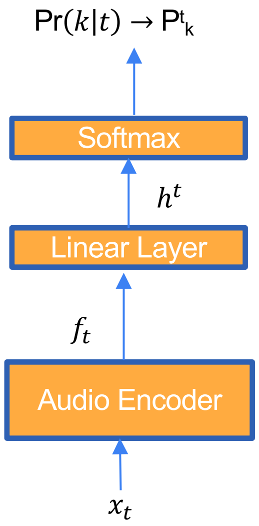
- Streaming

LAS:

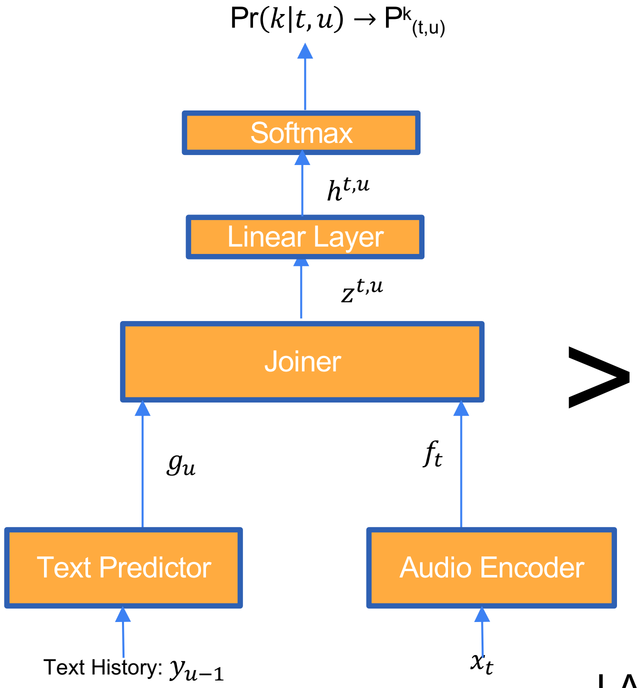
Attends to all audio embeddings and uses text history produced so far to generate probability distribution over output units.

$$f(y_u | y_{u-1}, y_{u-2}, \dots, y_0, h_0, h_2, h_3, \dots, h_v)$$

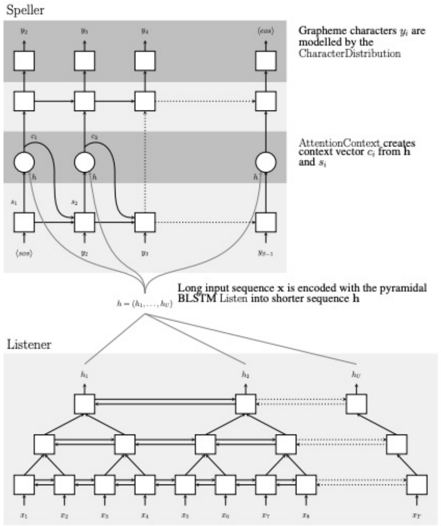
Conditional Independence Assumption



>



>



CTC:

- Only uses audio embedding at time "t" (x_t) to generate probability distribution over output units. CTC assumes that each output unit is conditionally independent of others.
- Streaming

RNN-T:

- Uses both audio embedding at time "t" (x_t) and text history produced so far (y_{u-1}) to generate probability distribution over output units.
- Streaming

LAS:

Attends to all audio embeddings and uses text history produced so far to generate probability distribution over output units.

$$f(y_u|y_{u-1}, y_{u-2}, ..., y_0, h_0, h_2, h_3, ..., h_v)$$

Why RNN-T ASR From Production Point Of View

- Single deployable non modularized neural model, all components of ASR in one model.
- Allows compact on-device streaming ASR. Does not need a decoder graph which can be large. Unlimited words in vocab. Standard on-device ASR choice across industry.

Google AI Blog

The latest news from Google AI

An All-Neural On-Device Speech Recognizer

Tuesday, March 12, 2019

Posted by Johan Schalkwyk, Google Fellow, Speech Team

Image from https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html

- Achieves comparable accuracy and compute with much smaller size model compared to modularized (hybrid) systems for production when training data is the same.

Table 3. *Comparison of Hybrid model with RNN-T*

Test Set	System	WER	Throughput	rtf@40
vid-clean	hybrid	14.0	55	.70
vid-clean	RNN-T	14.0	63	.60
vid-noisy	hybrid	20.7	55	.71
vid-noisy	RNN-T	21.0	65	.60

Example Study

Image from Jain et al., "RNN-T For Latency Controlled ASR With Imoroved Beam Search"

Components Of RNN-T ASR System

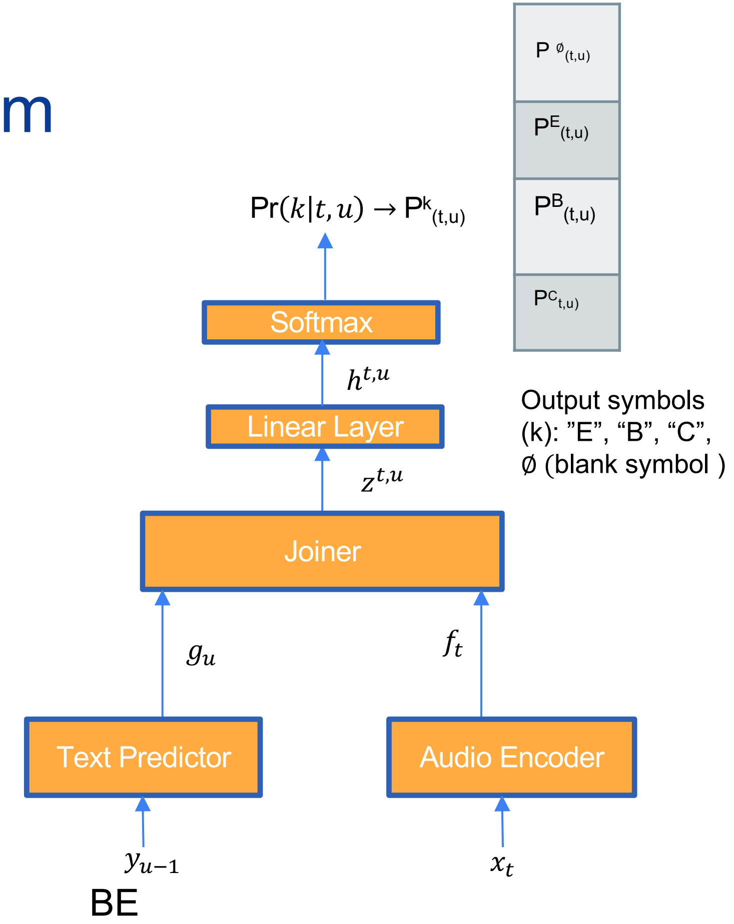
- Audio Encoder : To encode sequence of audio features into audio embeddings. Long short-term memory (LSTM), B-LSTM (bi-directional LSTM), Transformer are commonly used audio encoders. Acts as acoustic model

- Text Predictor : To encode transcript produced so far into text embedding. Typically a LSTM. Acts as language model

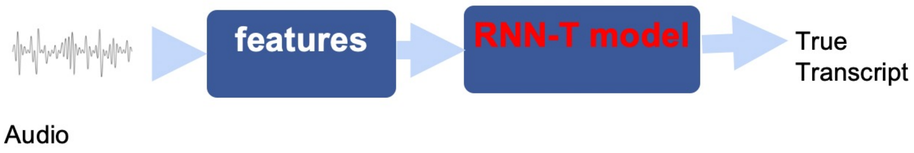
- Joiner combines output of audio encoder and text predictor

- A linear layer followed by Softmax produces probability distribution over output units. $\Pr(k|t,u)$($P^k_{(t,u)}$) is probability of emitting "k" from (t, u).

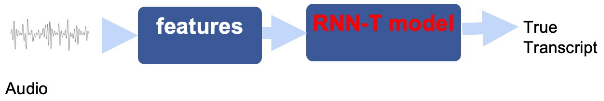
- No collapsing of symbols: If emitted output symbols is blank ($\emptyset$) then move to next time frame else stay in same time frame. $\emptyset$ also indicates "emit nothing"



Training



Training Objective



- Training utterance:
  “Input Audio” -> “BEE” (transcript)

- #output units (u): 3
  BEE -> *[[B], [E] ,[E]]*
- *#audio frames (t) : 4*

- We don’t know alignment i.e. which portion of audio aligns to what output unit (A path taken in lattice)

- Training Objective

$$P(BEE|X) = \sum_{alignment} P(alignment, BEE|X)$$

$$\sum_{alignment} P(BEE|alignment, X) P(alignment|X)$$

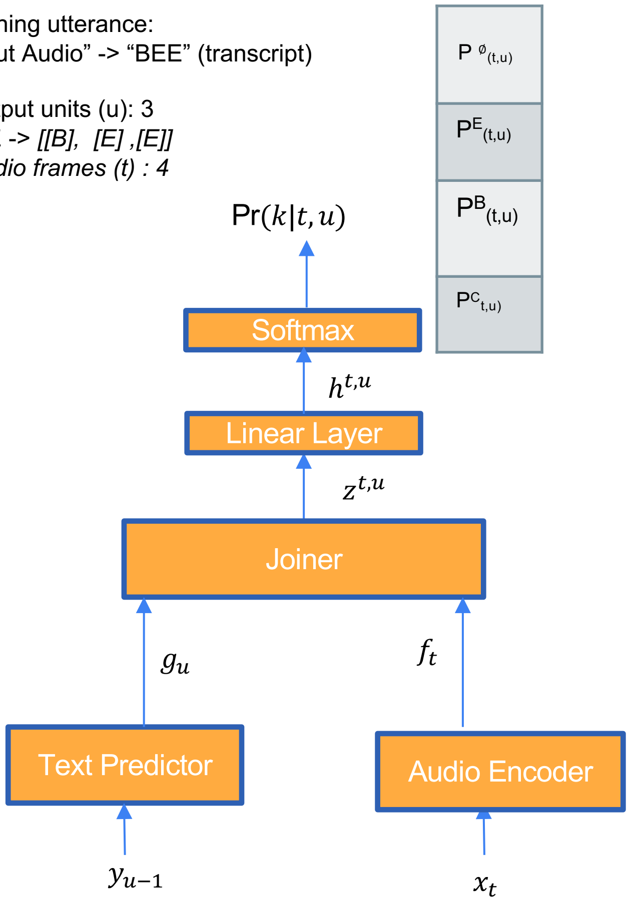
$$P(BEE|alignment, X) = 1 \qquad \sum_{alignment} P(alignment|X)$$

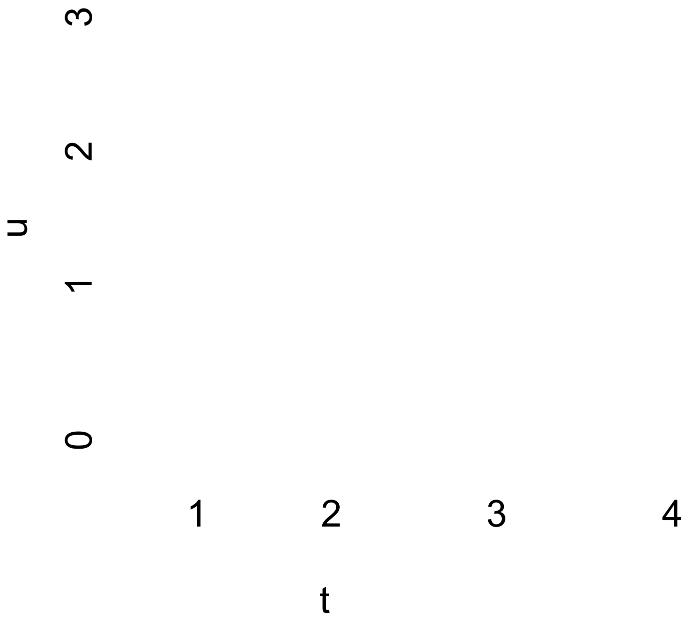
Graves et al., “[Sequence transduction with recurrent neural networks]”

Training

- Training utterance:
  “Input Audio” -> “BEE” (transcript)

- #output units (u): 3
  BEE -> *[[B], [E] ,[E]]*
- *#audio frames (t) : 4*





We don’t know alignment i.e. which portion of audio aligns to what output unit

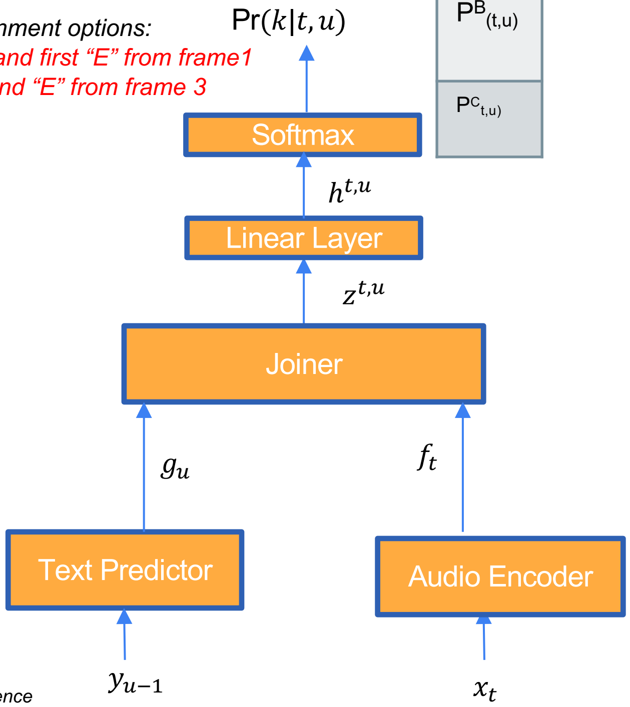
Graves et al., “[Sequence transduction with recurrent neural networks](#)”

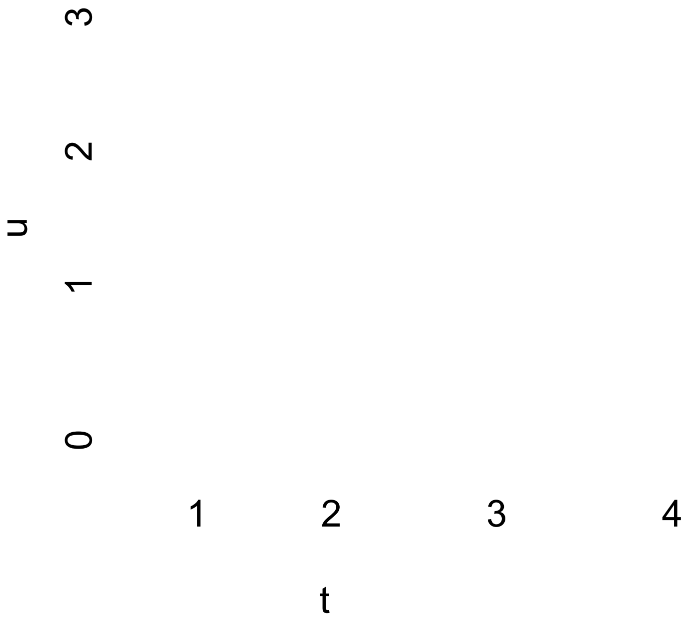
An Alignment During Training

- Training utterance:
  “Input Audio” -> “BEE” (transcript)

- #output units (u): 3
  BEE -> *[[B], [E] ,[E]]*
- *#audio frames (t) : 4*

- *Alignment options:*
- *“B” and first “E” from frame1 and second “E” from frame 3*





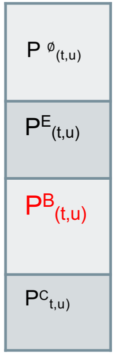
<BOS>: Begin of sentence

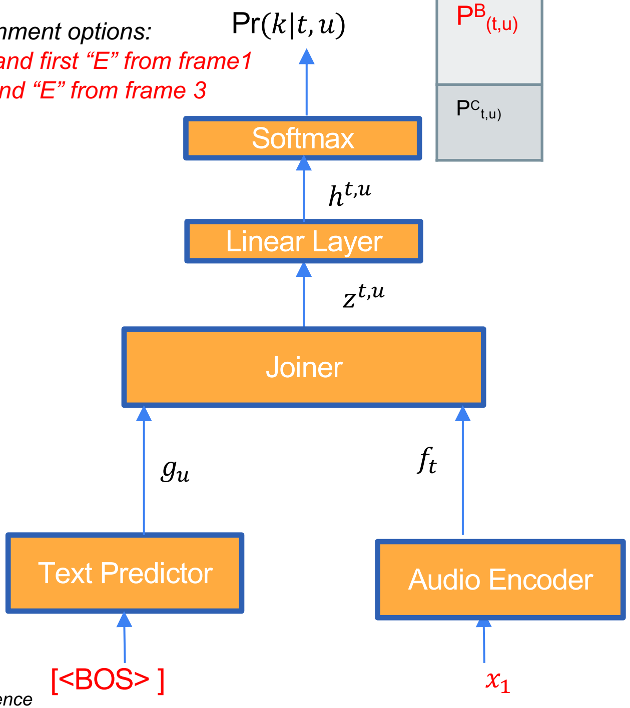
Graves et al., “Sequence transduction with recurrent neural networks”

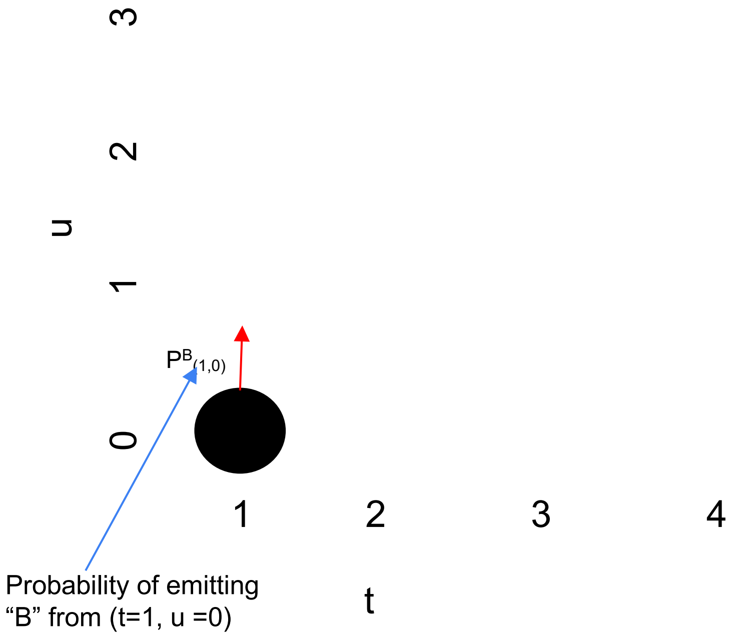
An Alignment During Training

If emitted output symbols is blank (∅) then move to next time frame else stay in same time frame. ∅ also indicates “emit nothing”

- Training utterance:
  “Input Audio” -> “BEE” (transcript)
- #output units (u): 3
  BEE -> *[[B], [E] ,[E]]*
- *#audio frames (t) : 4*
- *Alignment options:*
- *“B” and first “E” from frame1 and second “E” from frame 3*







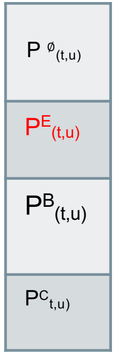
<BOS>: Begin of sentence

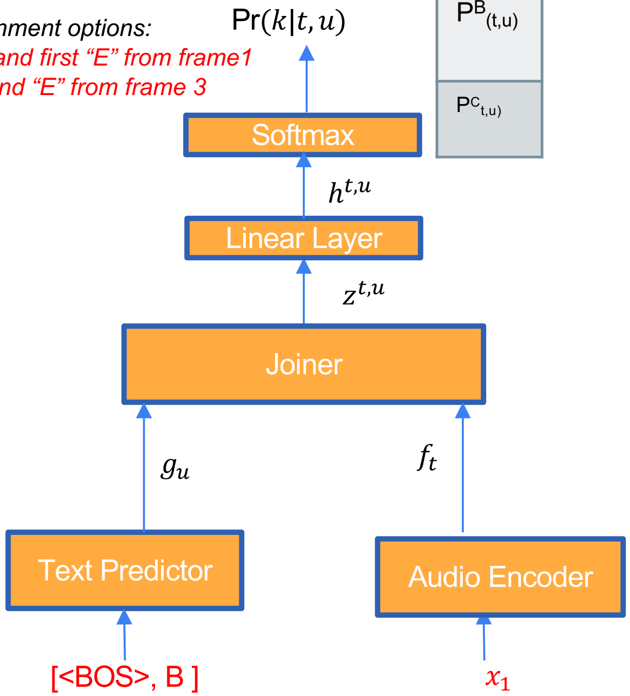
Graves et al., “Sequence transduction with recurrent neural networks”

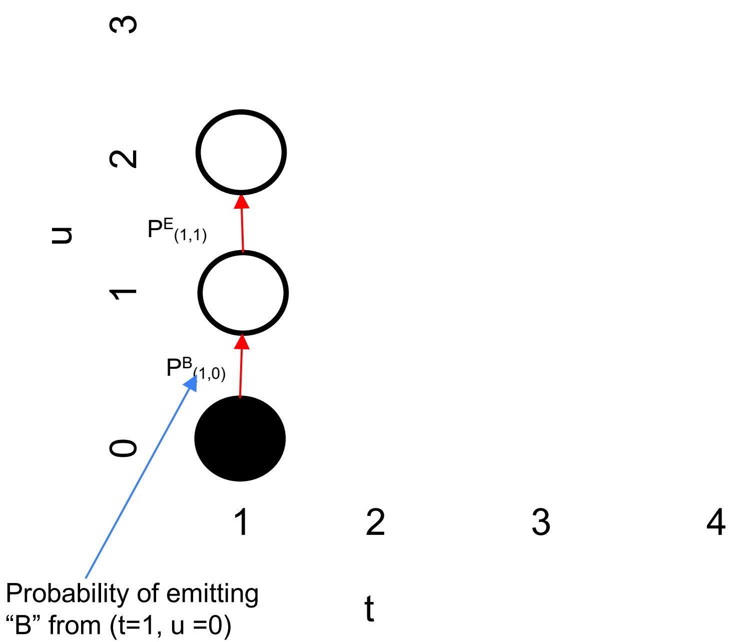
An Alignment During Training

If emitted output symbols is blank (∅) then move to next time frame else stay in same time frame. ∅ also indicates “emit nothing”

- Training utterance:
  “Input Audio” -> “BEE” (transcript)
- #output units (u): 3
  BEE -> *[[B], [E] ,[E]]*
- *#audio frames (t) : 4*
- *Alignment options:*
- *“B” and first “E” from frame1 and second “E” from frame 3*





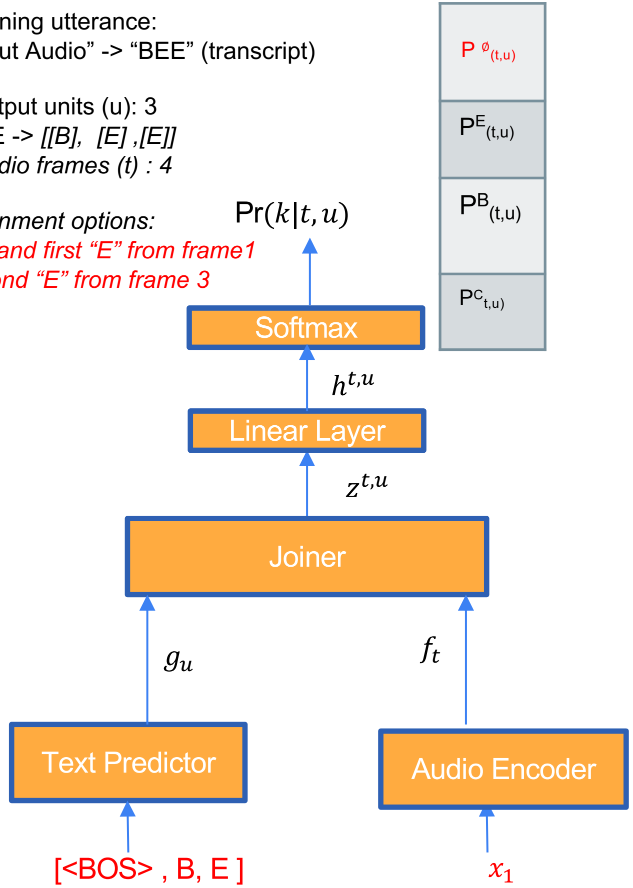


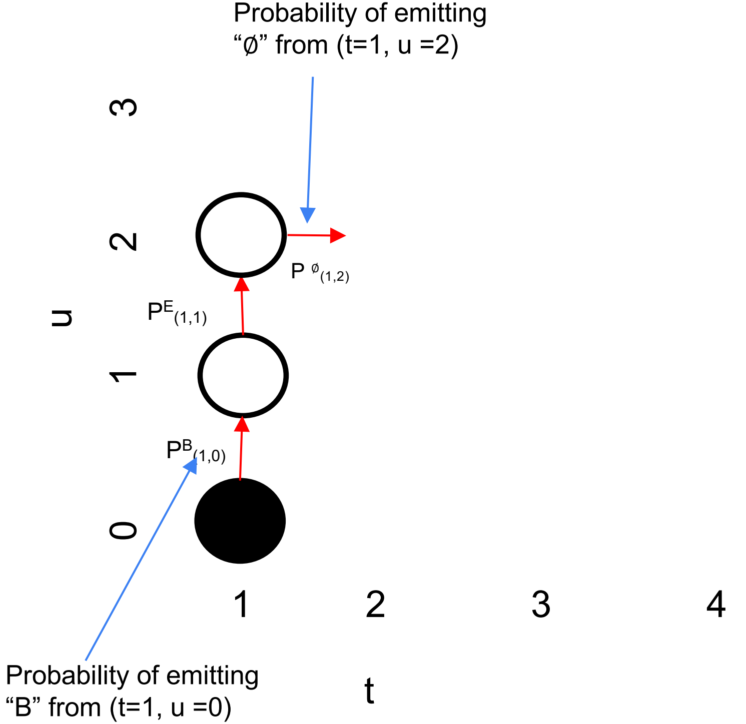
Graves et al., “Sequence transduction with recurrent neural networks”

An Alignment During Training

If emitted output symbols is blank (∅) then move to next time frame else stay in same time frame. ∅ also indicates "emit nothing"

- Training utterance:
  "Input Audio" -> "BEE" (transcript)
- #output units (u): 3
  BEE -> *[[B], [E] ,[E]]*
- *#audio frames (t) : 4*
- *Alignment options:*
- *"B" and first "E" from frame1 and second "E" from frame 3*



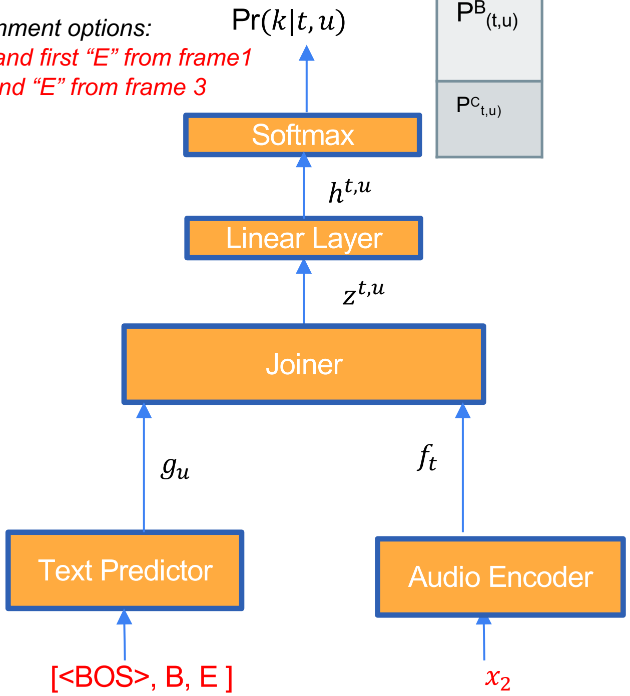


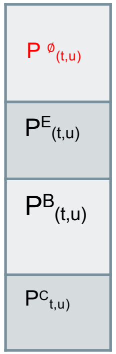
Graves et al., "Sequence transduction with recurrent neural networks"

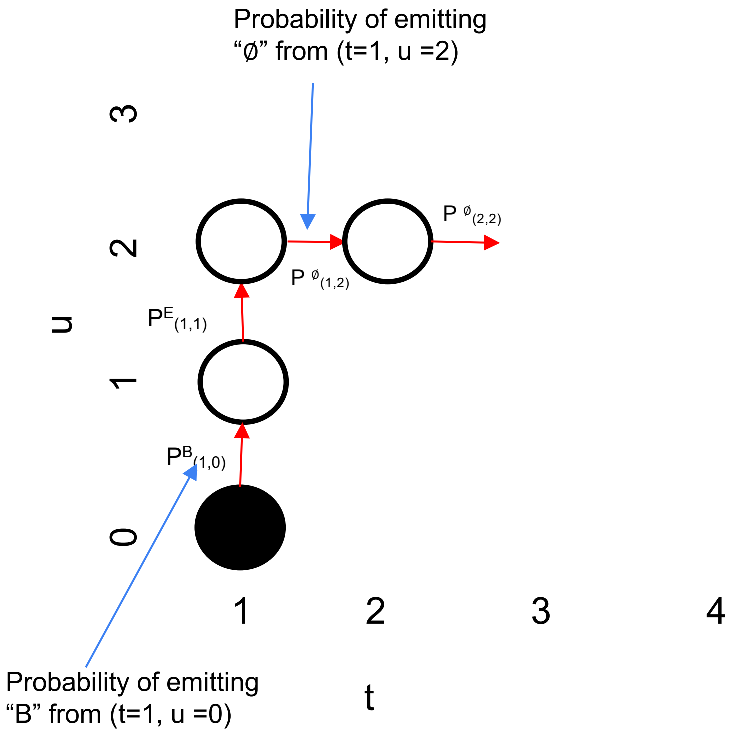
An Alignment During Training

If emitted output symbols is blank (∅) then move to next time frame else stay in same time frame. ∅ also indicates "emit nothing"

- Training utterance:
  "Input Audio" -> "BEE" (transcript)
- #output units (u): 3
  BEE -> *[[B], [E] ,[E]]*
- *#audio frames (t) : 4*
- *Alignment options:*
- *"B" and first "E" from frame1 and second "E" from frame 3*







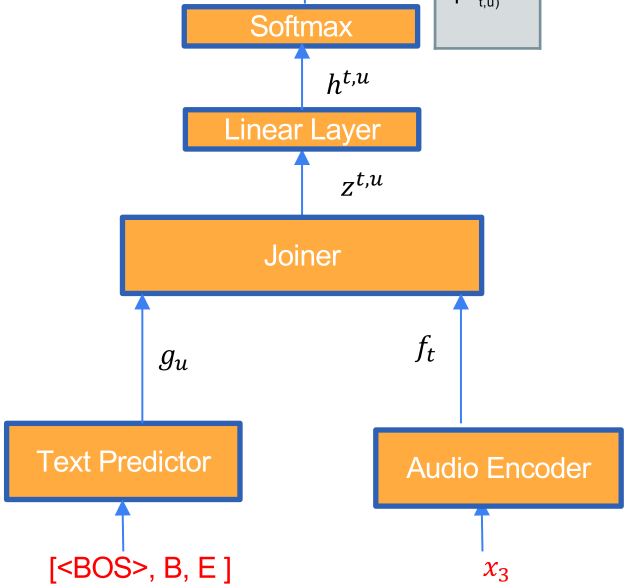
Graves et al., "Sequence transduction with recurrent neural networks"

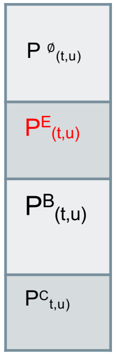
An Alignment During Training

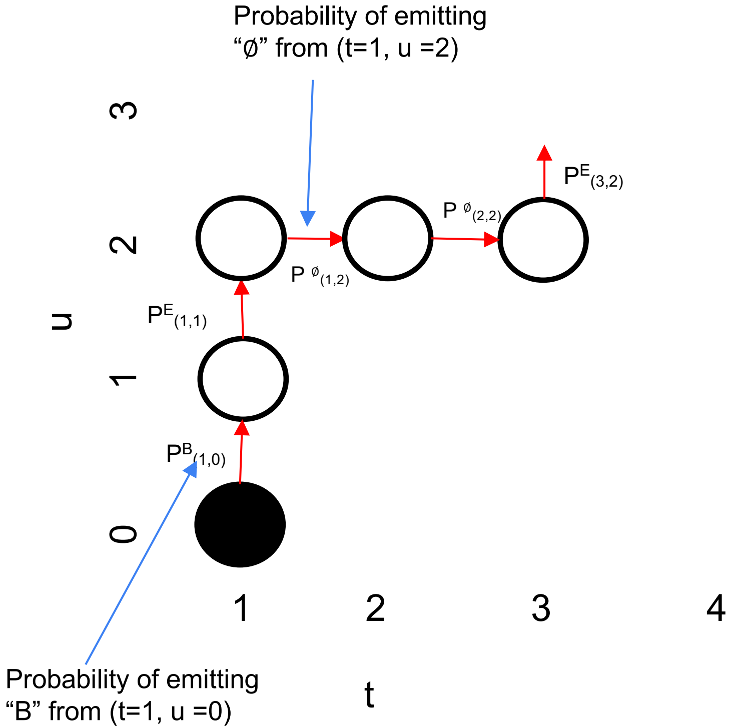
If emitted output symbols is blank (∅) then move to next time frame else stay in same time frame. ∅ also indicates "emit nothing"

- Training utterance:
  "Input Audio" -> "BEE" (transcript)
- #output units (u): 3
  BEE -> *[[B], [E] ,[E]]*
- *#audio frames (t) : 4*
- *Alignment options:*
- *"B" and first "E" from frame1 and second "E" from frame 3*

$\Pr(k|t,u)$





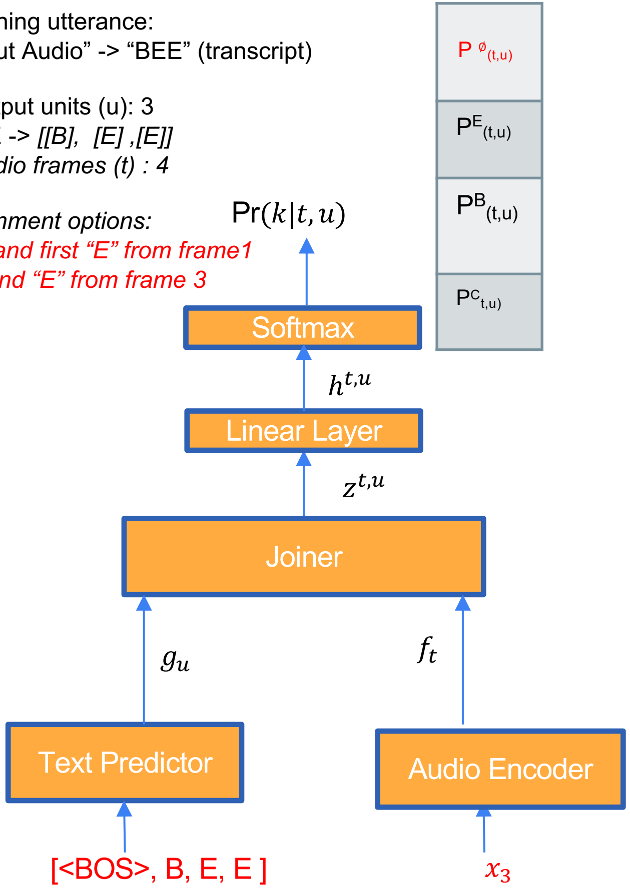


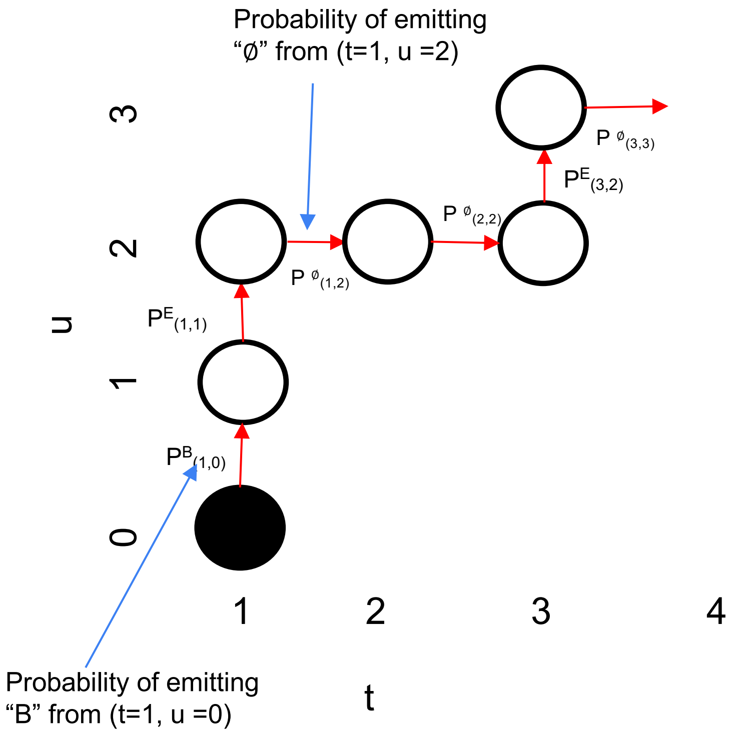
Graves et al., "Sequence transduction with recurrent neural networks"

An Alignment During Training

If emitted output symbols is blank (∅) then move to next time frame else stay in same time frame. ∅ also indicates "emit nothing"

- Training utterance:
  "Input Audio" -> "BEE" (transcript)
- #output units (u): 3
  BEE -> *[[B], [E] ,[E]]*
- *#audio frames (t) : 4*
- *Alignment options:*
- *"B" and first "E" from frame1 and second "E" from frame 3*



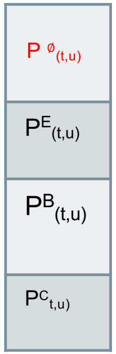


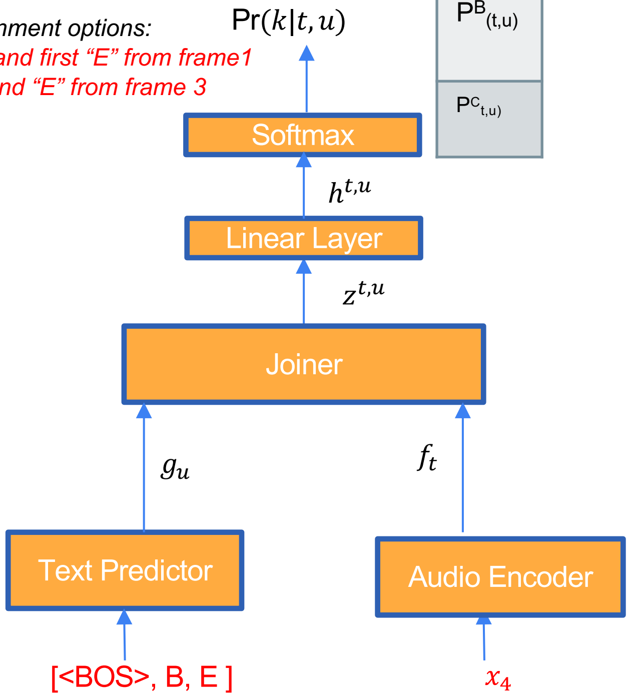
Graves et al., "[Sequence transduction with recurrent neural networks]"

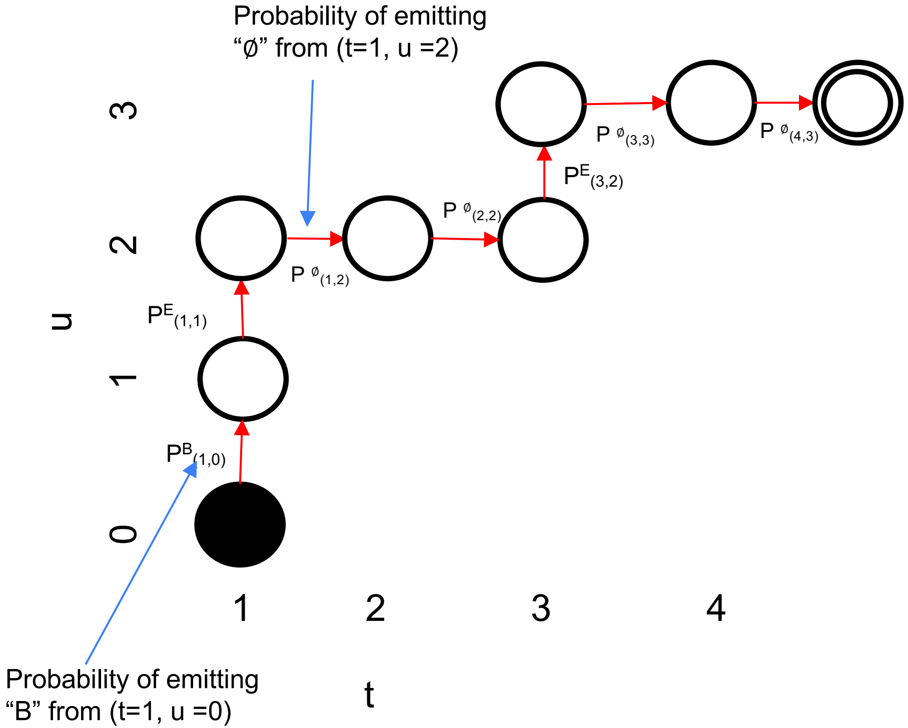
An Alignment During Training

If emitted output symbols is blank (∅) then move to next time frame else stay in same time frame. ∅ also indicates "emit nothing"

- Training utterance:
  "Input Audio" -> "BEE" (transcript)
- #output units (u): 3
  BEE -> *[[B], [E] ,[E]]*
- *#audio frames (t) : 4*
- *Alignment options:*
- *"B" and first "E" from frame1 and second "E" from frame 3*

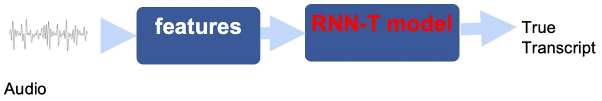




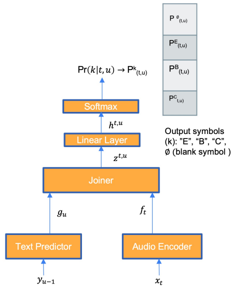


Graves et al., "Sequence transduction with recurrent neural networks"

RNN-T Lattice And Training



- Training utterance:
  "Input Audio" -> "BEE" (transcript)
- #output units (u): 3
  BEE -> *[[B], [E] ,[E]]*
- *#audio frames (t) : 4*
- We don't know alignment i.e. which portion of audio aligns to what output unit



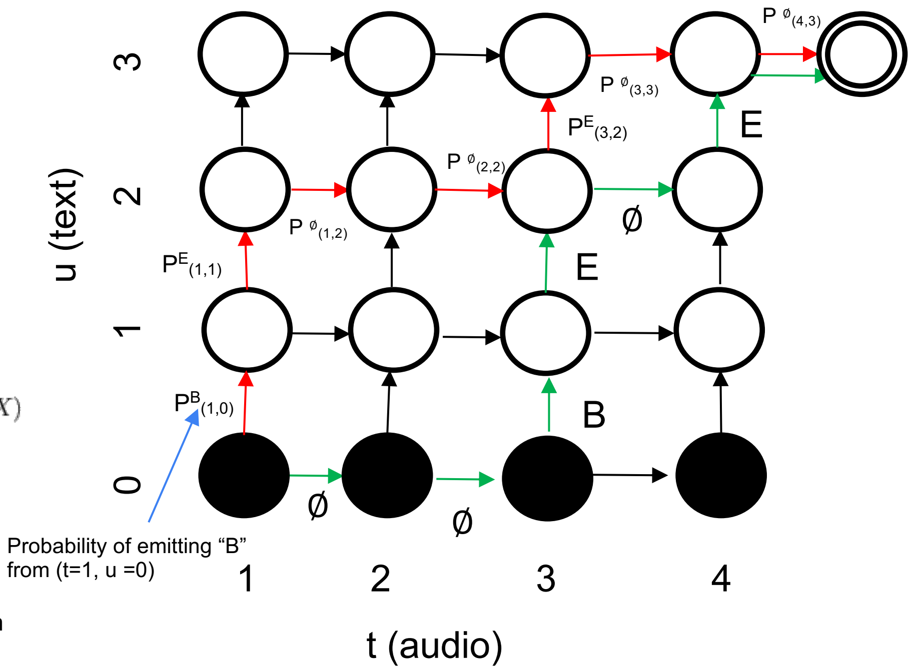
- Probability of alignment is multiplication of probabilities assigned along the path of alignment
- Training

$$P(BEE|X) = \sum_{alignment} P(alignment, BEE|X)$$

$$\sum_{alignment} P(BEE|alignment, X)P(alignment|X)$$

$$P(BEE|alignment, X) = 1$$

$$\sum_{alignment} P(alignment|X)$$



- Lattice contains all valid alignment paths(traversals). **During training, we change (optimize) neural network parameters to maximize sum of probabilities all alignment paths**
- **Computation is done efficiently through dynamic programming (slide 51 to 56)**

Graves et al., "Sequence transduction with recurrent neural networks"

Inference: Find a Transcript given an audio and Trained RNN-T model

Goal: Find candidate transcript of a given audio during test time.
Challenge: There are infinitely many alignments that can be assigned to a test audio.

Operation: Extend Hypothesis

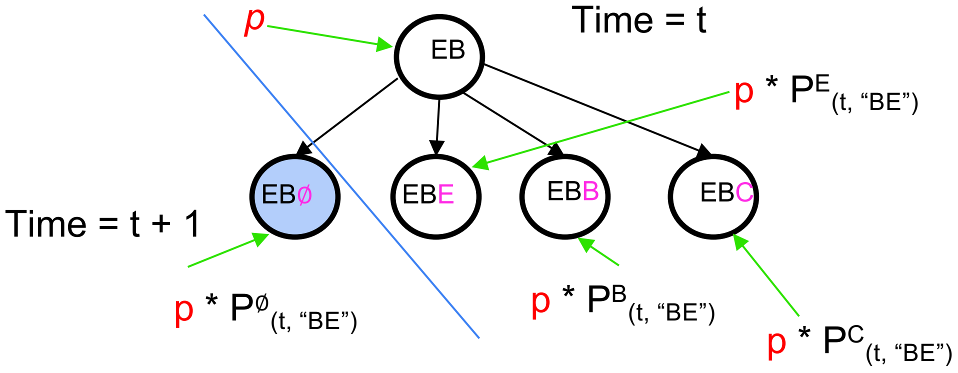
- Hypothesis is defined as a candidate output sequence during search

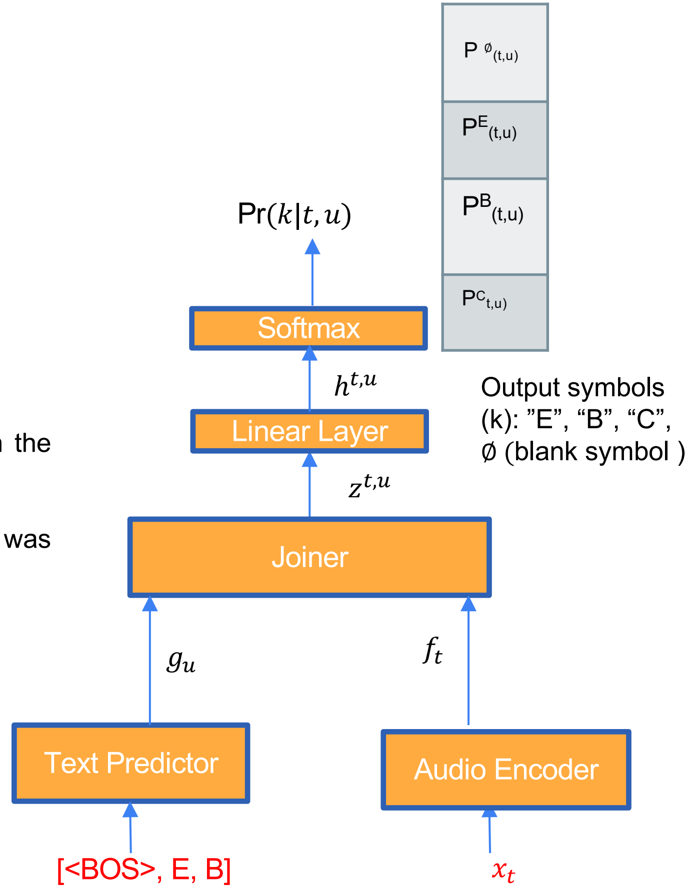
- Example Hypothesis: “EB” at time frame “t”
- Output symbols: ”E”, “B”, “C”, “∅”

- Extend Hypothesis: Append hypothesis with each of the output symbols (k) and ∅

- The ∅ extension goes to next time frame (t + 1) and non-blank extensions remain in the same time frame (t)

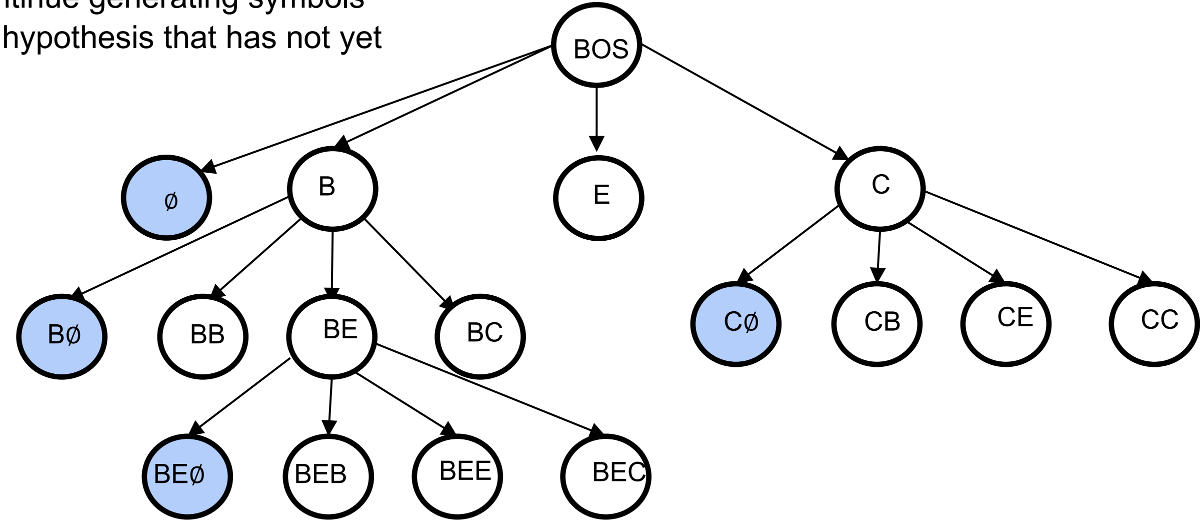
- Every extended hypothesis has lower probability compared to the hypothesis it was extended from

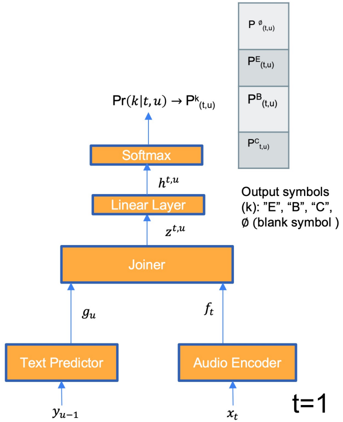


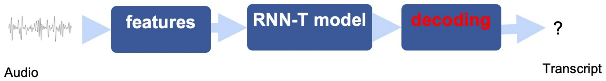


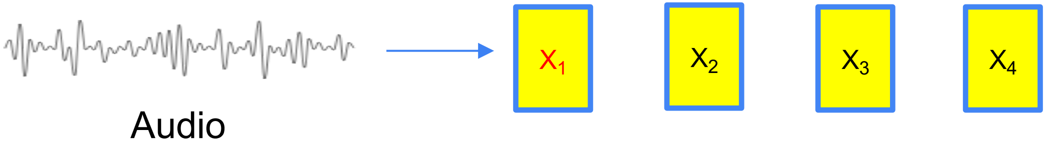
Subset Of Hypotheses At t = 1 During Decoding

- We can continue generating symbols from every hypothesis that has not yet emitted Ø



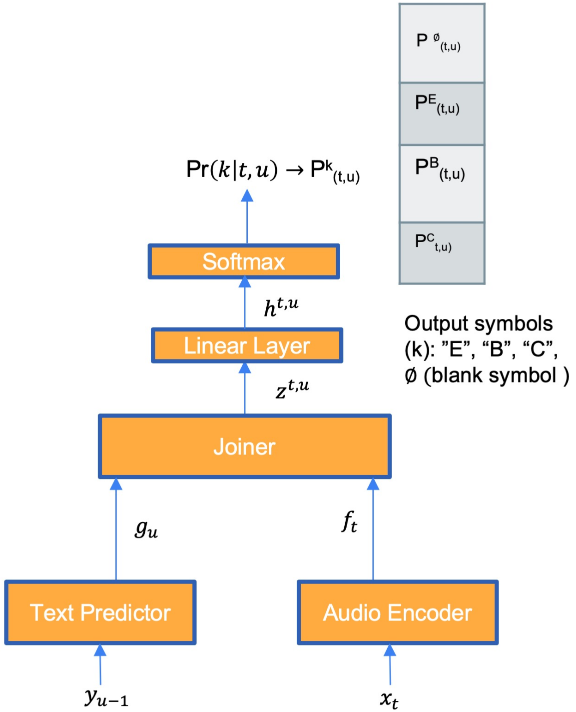






Greedy Decoding

- Goal: Get a candidate final transcript by making local optimal choice at each distinct “t” and “u”

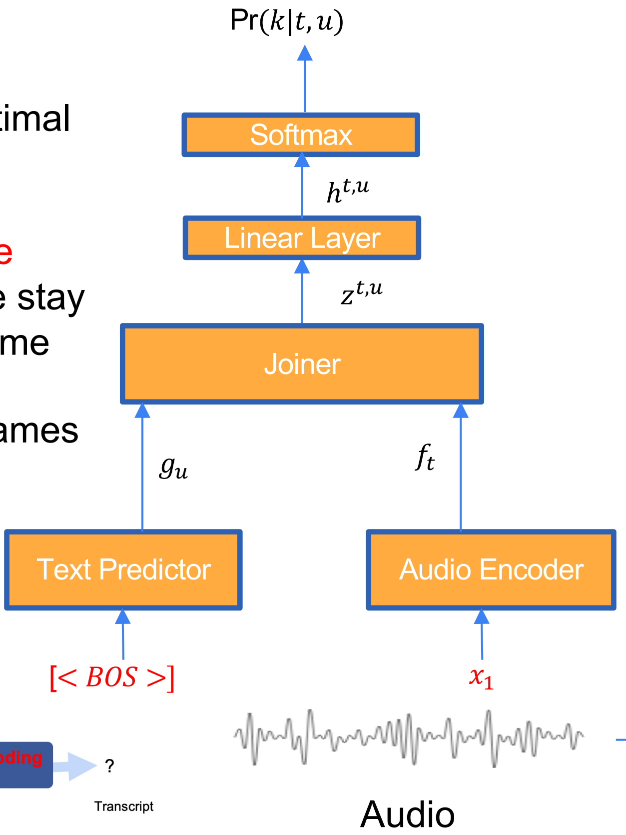


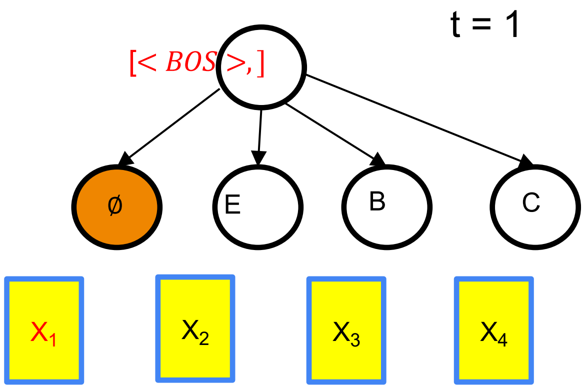
Greedy Decoding

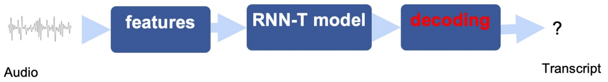
- Lets make a local optimal choice: If highest probability is given to blank (∅), move to the next audio frame else stay in the same audio frame

- Do it until all audio frames are processed

k	probability
∅	0.4
E	0.2
B	0.3
C	0.1



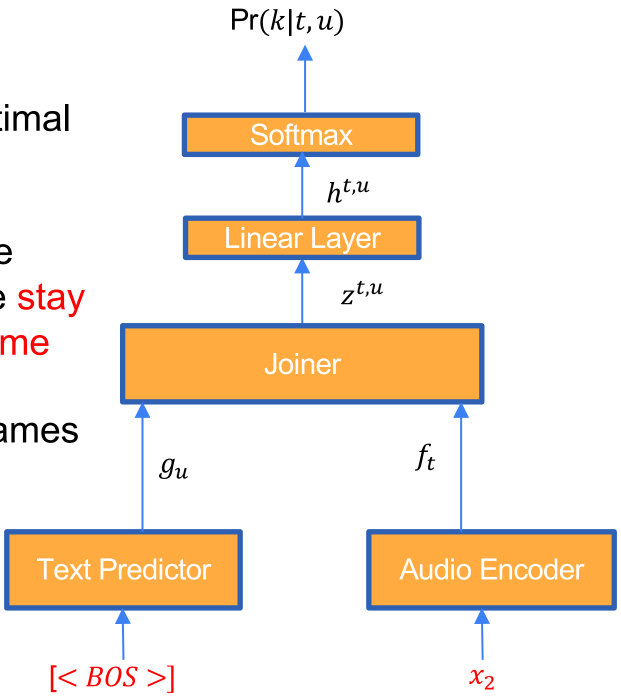


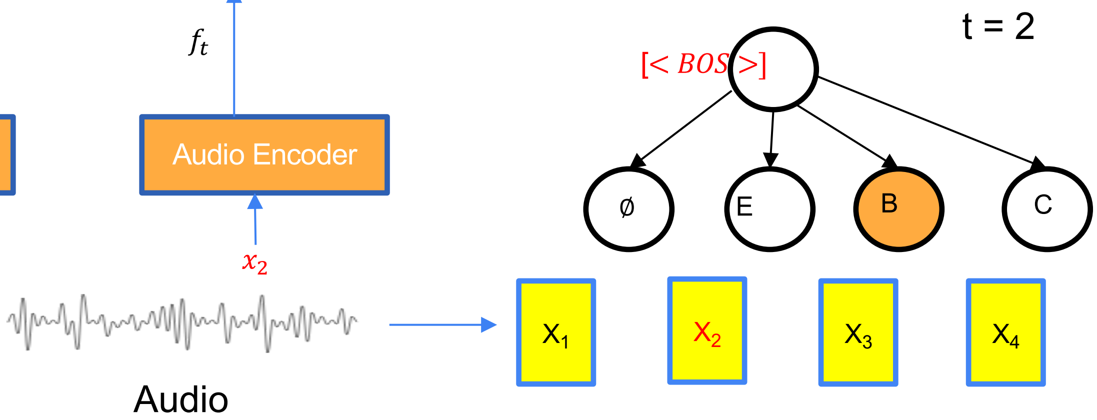


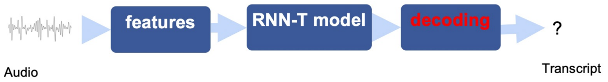
Greedy Decoding

k	probability
∅	0.04
E	0.04
B	0.3
C	0.01

- Lets make a local optimal choice: If highest probability is given to blank (∅), move to the next audio frame else stay in the same audio frame
- Do it until all audio frames are processed



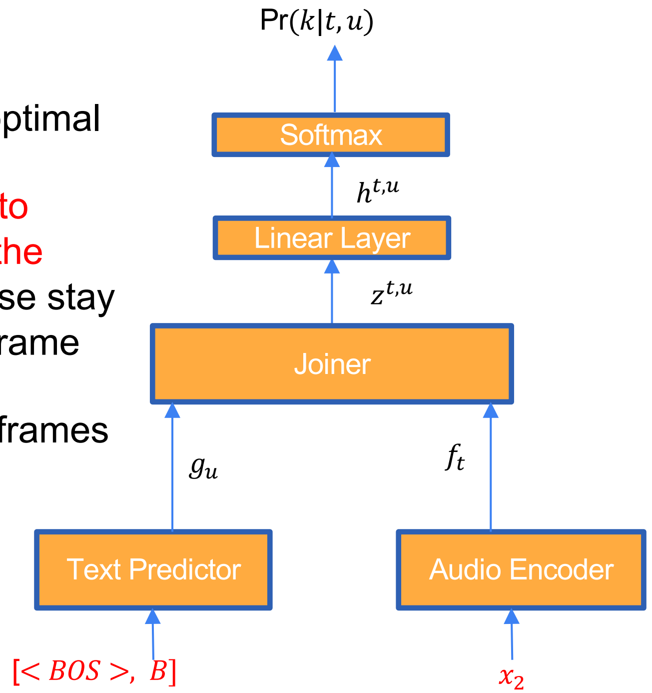


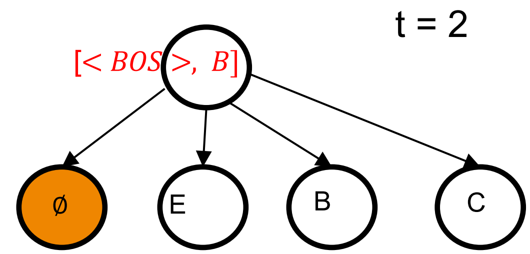


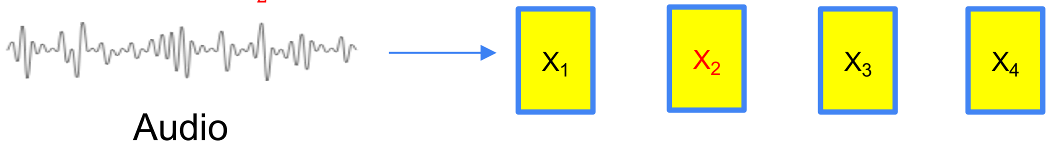
Greedy Decoding

- Lets make a local optimal choice: If highest probability is given to blank (∅), move to the next audio frame else stay in the same audio frame
- Do it until all audio frames are processed

k	probability
∅	0.4
E	0.1
B	0.2
C	0.3



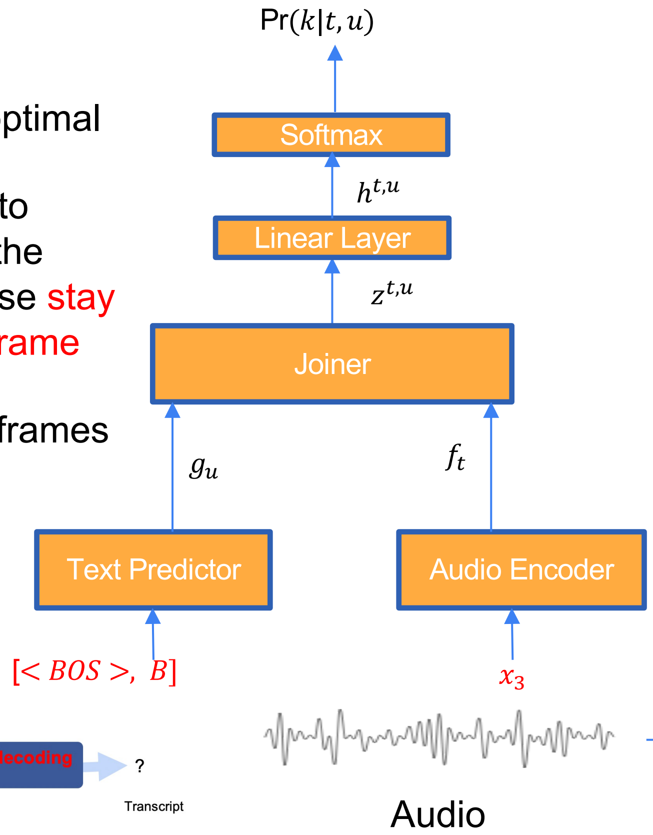


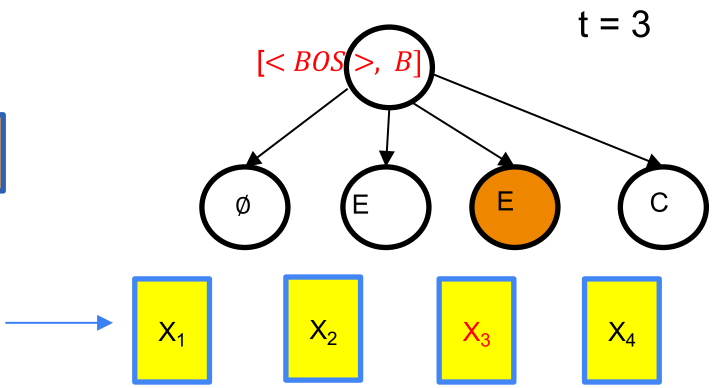


Greedy Decoding

k	probability
∅	0.3
E	0.35
B	0.3
C	0.15

- Lets make a local optimal choice: If highest probability is given to blank (∅), move to the next audio frame else stay in the same audio frame
- Do it until all audio frames are processed



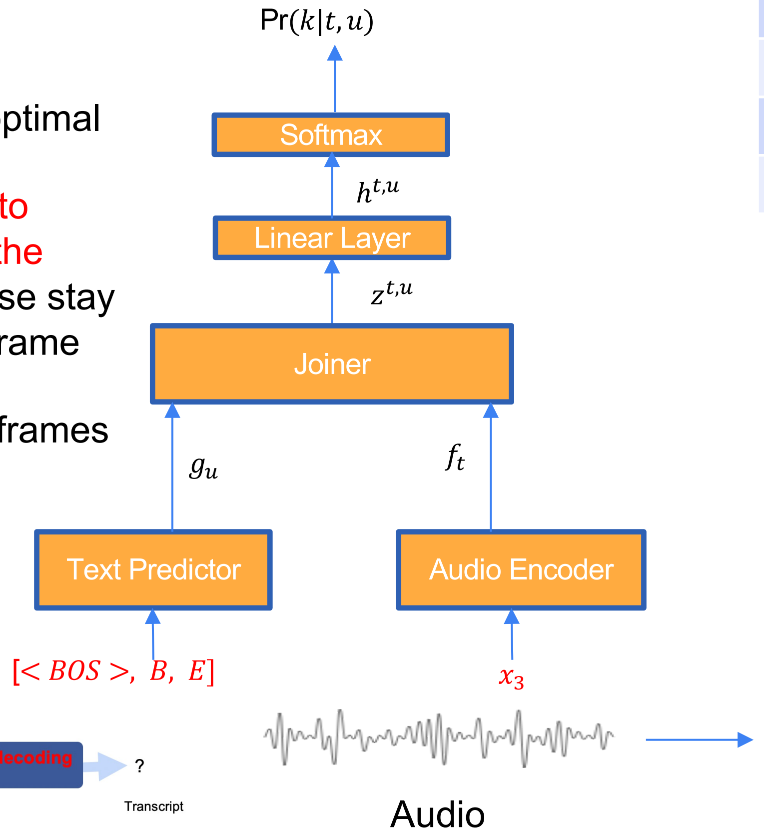


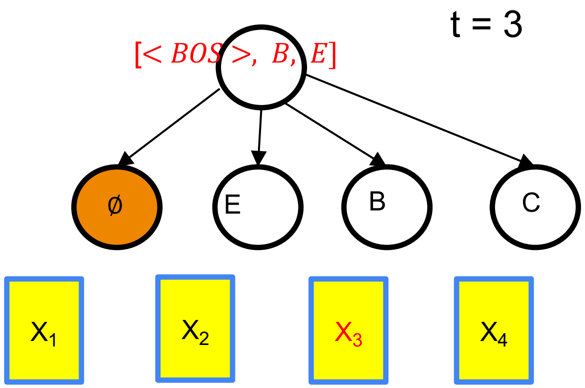
Greedy Decoding

- Lets make a local optimal choice: If highest probability is given to blank (Ø), move to the next audio frame else stay in the same audio frame

- Do it until all audio frames are processed

k	probability
Ø	0.34
E	0.26
B	0.25
C	0.25

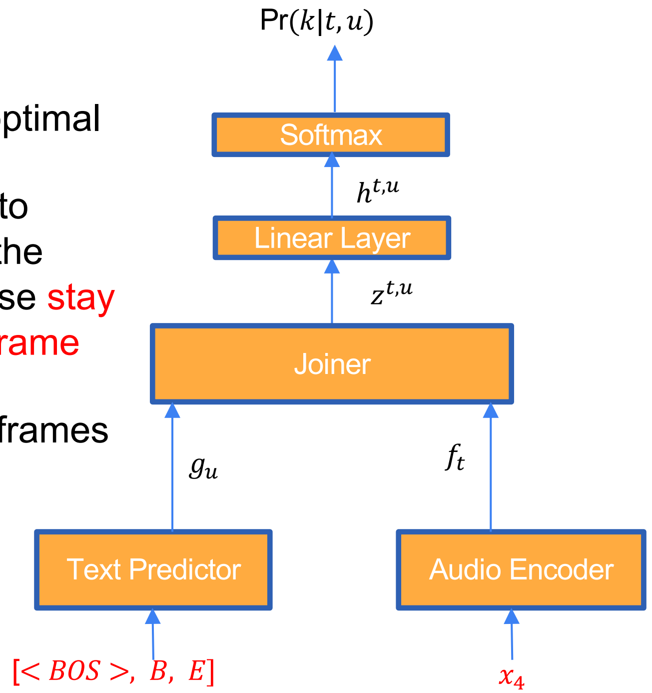




Greedy Decoding

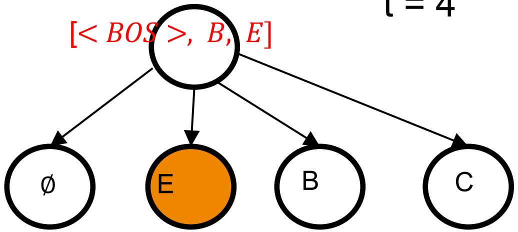
- Lets make a local optimal choice: If highest probability is given to blank (∅), move to the next audio frame else stay in the same audio frame

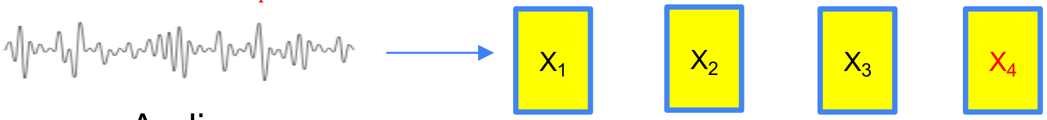
- Do it until all audio frames are processed



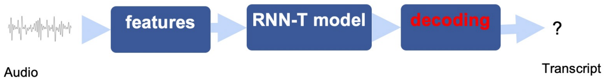
k	probability
∅	0.3
E	0.4
B	0.28
C	0.02

t = 4



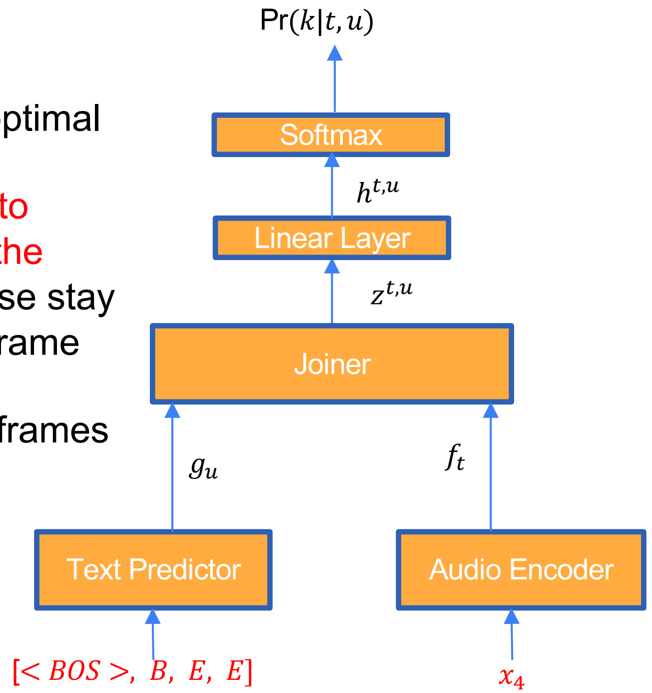


Audio

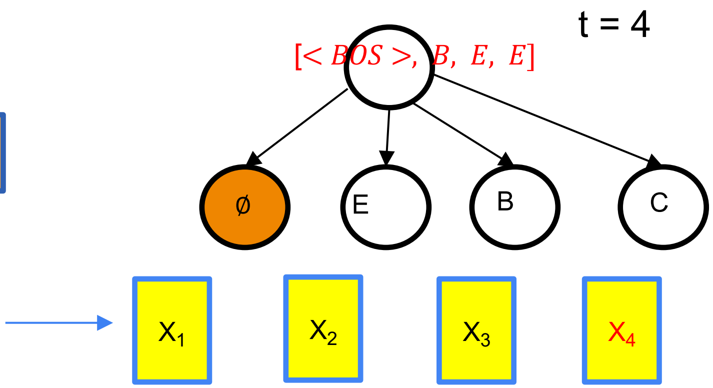


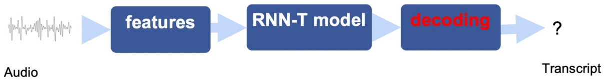
Greedy Decoding

- Lets make a local optimal choice: If highest probability is given to blank (∅), move to the next audio frame else stay in the same audio frame
- Do it until all audio frames are processed



k	probability
∅	0.4
E	0.25
B	0.3
C	0.05

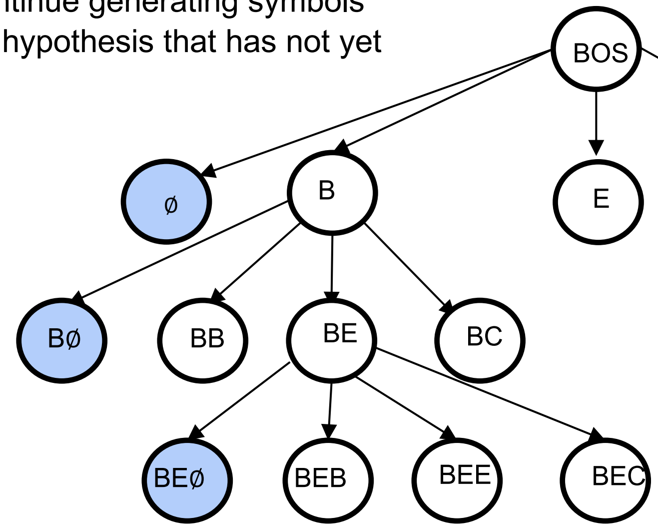


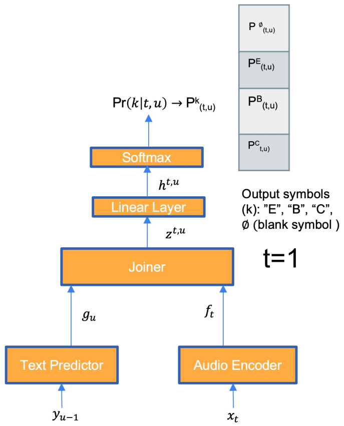


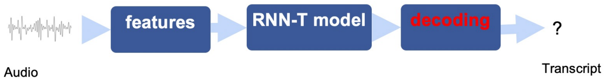
Audio

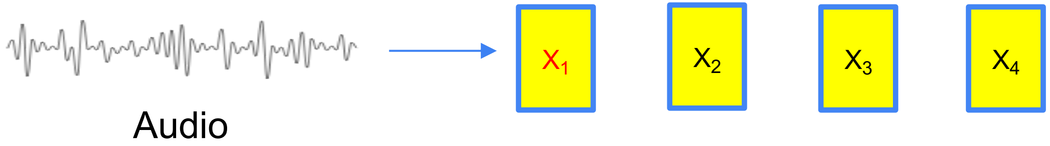
Subset Of Hypotheses At t = 1 During Decoding

- We can continue generating symbols from every hypothesis that has not yet emitted ∅









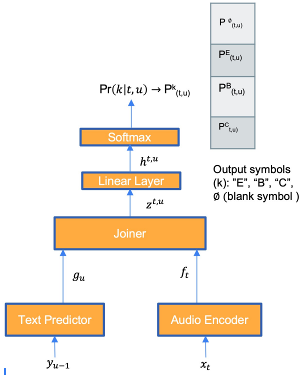
Beam Search

- We don’t want to stay in time frame “t” forever. But we do not know when to move to “t+1” as there are infinitely many candidates that can be explored in time frame “t”.

- Goal: We would like to obtain *n* candidate hypotheses at time frame “t+1” before exiting to decode at “t” which would be better than all possible future extensions at “t” that could go to “t+1”.

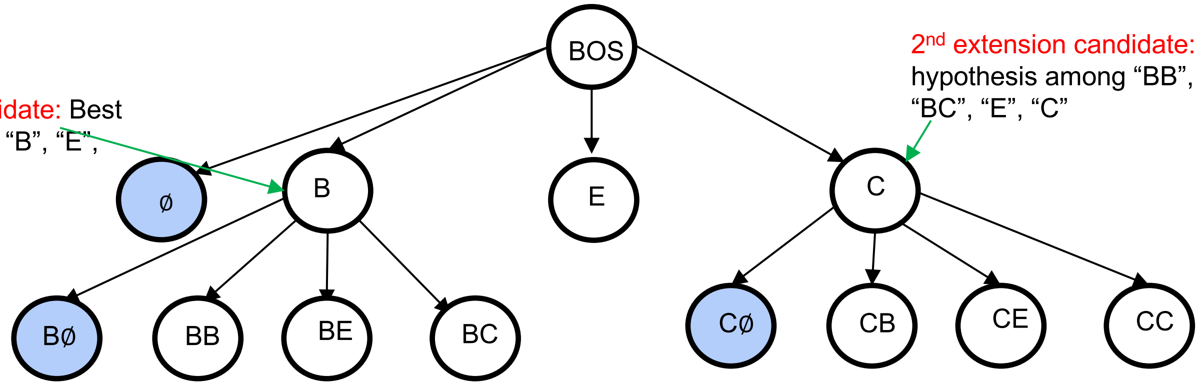
- *n* is hyper parameter

Beam Search: Expand the best hypothesis among candidate hypotheses



1st extension candidate: Best hypothesis among “B”, “E”, “C”

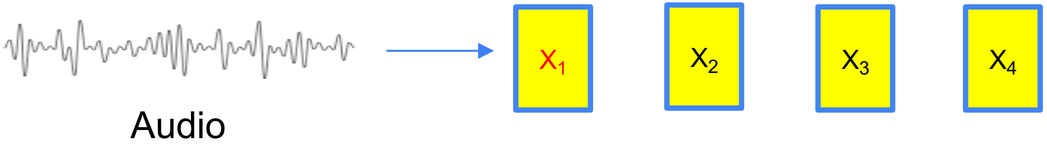
2nd extension candidate: Best hypothesis among “BB”, “BE”, “BC”, “E”, “C”



BØ
CØ
Ø

Hypothesis that has emitted Ø and hence gone to “t+1”. Beam Size (n) “3”

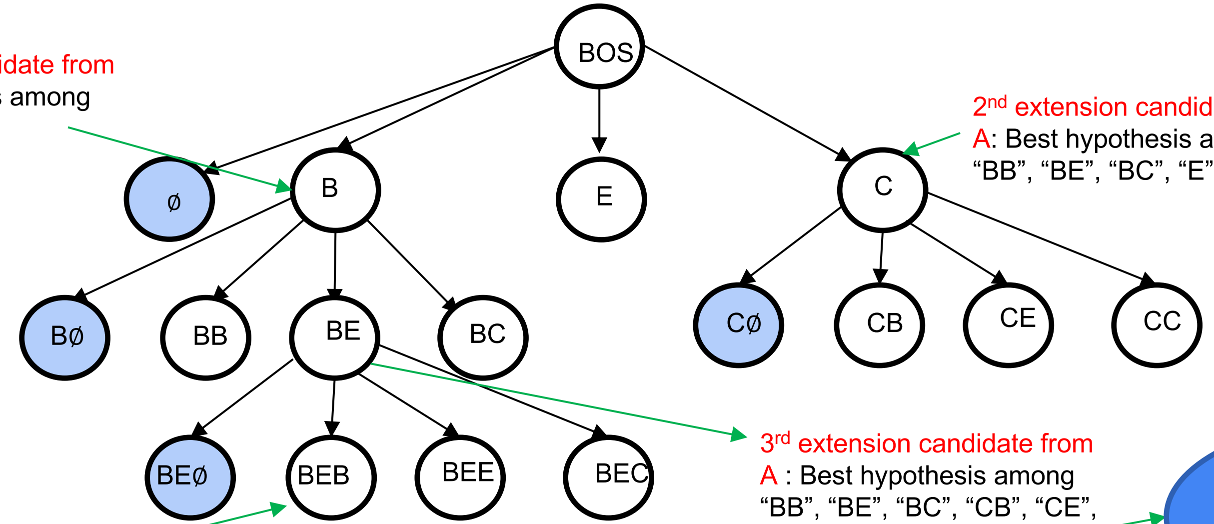
Is beam size (n) number of extensions enough? No.



Beam Search: When to exit search at time frame t

- Set B: Hypotheses that has emitted ∅ from frame "t" and are now in "t+1"
- Set A: Hypotheses that has not yet emitted ∅ from frame "t" and so can continue emitting more symbols from "t"

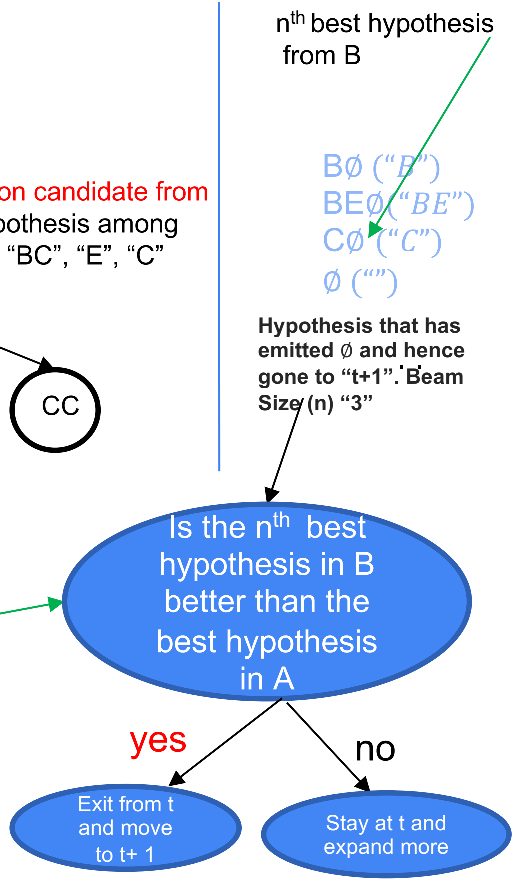
1st extension candidate from A: Best hypothesis among "B", "E", "C"



2nd extension candidate from A: Best hypothesis among "BB", "BE", "BC", "E", "C"

3rd extension candidate from A : Best hypothesis among "BB", "BE", "BC", "CB", "CE", "CC"

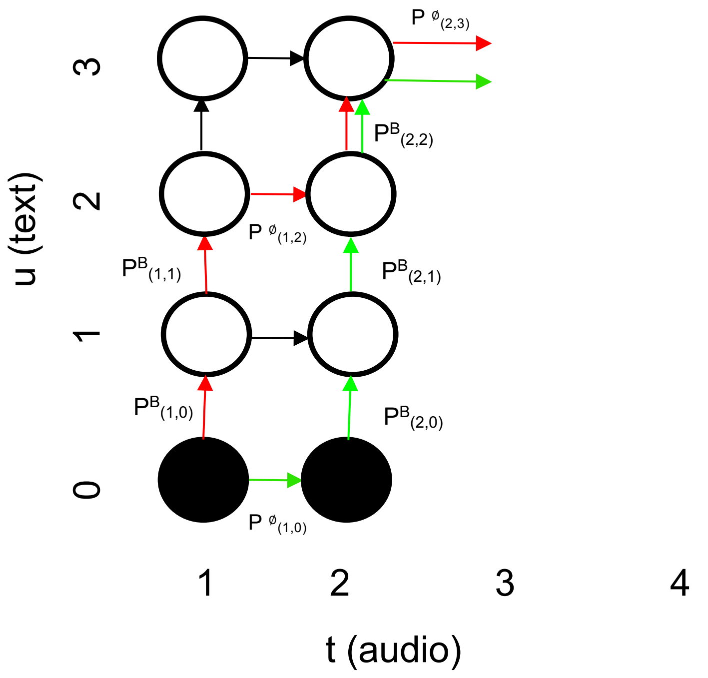
4th extension candidate from A : Best hypothesis among "BB", "BEB", "BEE", "BEC", "BC", "CB", "CE", "CC"



Recap: Every extended hypothesis has lower probability compared to hypothesis it was extended from.

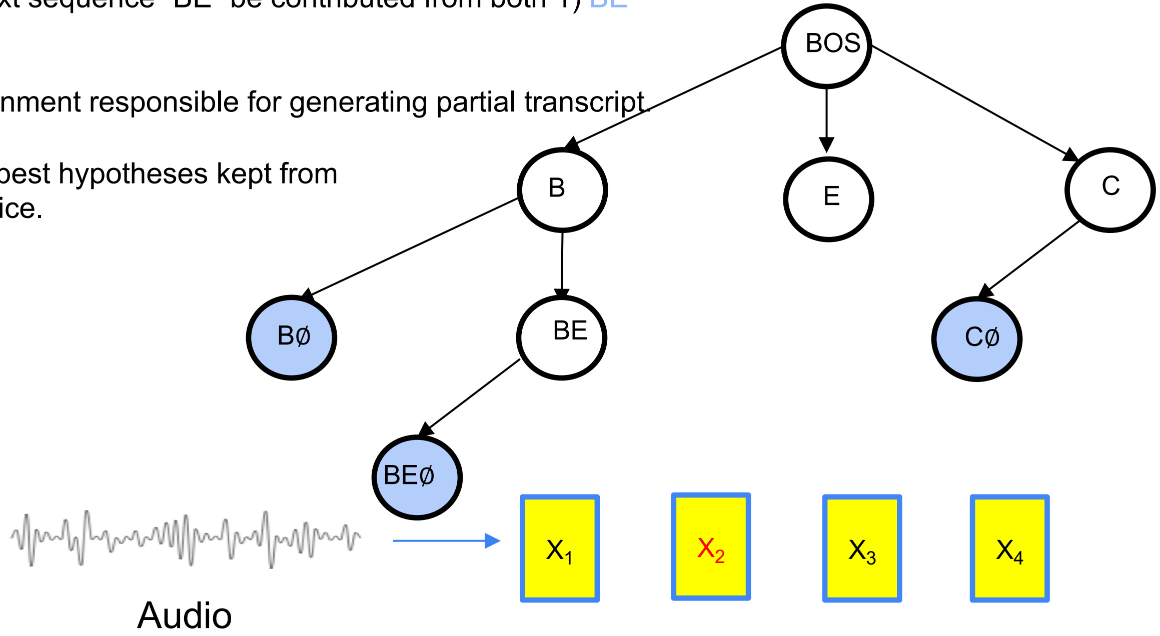
Beam Search: How To Deal With hypotheses that results in same partial transcript in B

- Combine probabilities of different alignments that result in same partial transcript
  - BB ∅ B ∅ (First frame emitted two Bs and Second Frame emitted one B)
  - ∅ BBB ∅ (First frame emitted nothing and Second Frame emitted three Bs)
  - The total probability of “BBB” at exit of time index “2” would be some of probability of alignments : BB ∅ B ∅ and ∅ BBB ∅
    - These are two different alignments that result in “BBB” at t=2.



How To Deal With Hypotheses That Share Prefix

- The three best hypotheses from previous time frame corresponds to following partial transcripts : B & BE & C
- RNN-T allows emission of any number of non blank symbols from a time frame and as hypothesis “B” from previous time frame can be extended to “BE” by emitting “E” at the current time frame, should the total probability of text sequence “BE” be contributed from both 1) BE and 2) B by emitting “E”? Yes!!
- Add prefix accumulation for each of the alignment responsible for generating partial transcript.
- We are limiting prefix accumulation to only best hypotheses kept from previous time frame, which works well in practice.



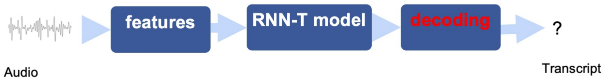
Beam Search: Algorithm

- Set B: Hypotheses that a blank has been output from frame ***t***
- Set A: Hypotheses that a blank has not been output from frame ***t***
- Take the best hypothesis from A and extend it with each of the output symbols and ∅
- Exit the beam search at audio frame ***t*** if B has more than W (beam size) hypotheses that are more probable than most probable hypothesis in A.
- When starting the beam search at audio frame ***t+1***: 1) Empty A, 2) Move all Hypotheses from B to A, 3) compute the prefix (*pref*) completion probability and, 4) do prefix accumulation.
- Prefix completion probability (**Pr(y|$\hat{y}$,t))** for proper prefixes($\hat{y}$ ∈ *pref*(y)) of each hypothesis (y) is computed by outputting the symbols of symbol index (u) from (|$\hat{y}$| + 1) to (|y|) at audio frame (t). (slide 60 has more context about it)

$$\Pr(\boldsymbol{y}|\hat{\boldsymbol{y}}, t) = \prod_{u=|\hat{\boldsymbol{y}}|+1}^{|\boldsymbol{y}|} \Pr(y_u|\boldsymbol{y}_{[0:u-1]}, t)$$

- Prefix accumulation for each hypothesis in A is done by following:

for $\boldsymbol{y}$ **in** A **do**
$\quad \Pr(\boldsymbol{y}) \mathrel{+}= \sum_{\hat{\boldsymbol{y}} \in pref(\boldsymbol{y}) \cap A} \Pr(\hat{\boldsymbol{y}}) \Pr(\boldsymbol{y}|\hat{\boldsymbol{y}}, t)$
end for



Algorithm 1 Output Sequence Beam Search

Initalise: $B = \{\varnothing\}$; $\Pr(\varnothing) = 1$
for $t = 1$ **to** T **do**
$\quad A = B$
$\quad B = \{\}$
$\quad$ **for** $\boldsymbol{y}$ **in** A **do**
$\quad\quad \Pr(\boldsymbol{y}) \mathrel{+}= \sum_{\hat{\boldsymbol{y}} \in pref(\boldsymbol{y}) \cap A} \Pr(\hat{\boldsymbol{y}}) \Pr(\boldsymbol{y}|\hat{\boldsymbol{y}}, t)$
$\quad$ **end for**
$\quad$ **while** B contains less than W elements more probable than the most probable in A **do**
$\quad\quad \boldsymbol{y}^* =$ most probable in A
$\quad\quad$ Remove $\boldsymbol{y}^*$ from A
$\quad\quad \Pr(\boldsymbol{y}^*) = \Pr(\boldsymbol{y}^*) \Pr(\varnothing|\boldsymbol{y}, t)$
$\quad\quad$ Add $\boldsymbol{y}^*$ to B
$\quad\quad$ **for** $k \in \mathcal{Y}$ **do**
$\quad\quad\quad \Pr(\boldsymbol{y}^* + k) = \Pr(\boldsymbol{y}^*) \Pr(k|\boldsymbol{y}^*, t)$
$\quad\quad\quad$ Add $\boldsymbol{y}^* + k$ to A
$\quad\quad$ **end for**
$\quad$ **end while**
$\quad$ Remove all but the W most probable from B
end for
Return: $\boldsymbol{y}$ with highest $\log \Pr(\boldsymbol{y})/|\boldsymbol{y}|$ in B

Graves et al., "[Sequence transduction with recurrent neural networks]"

Practical Considerations

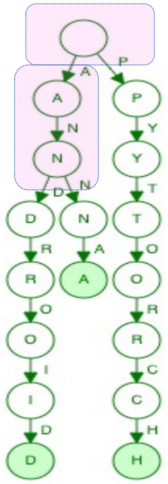
Utilizing Utterance Specific Context: Contextualization

- Use utterance specific list of context words along with audio during RNN-T ASR training and/or inference
- Improve recognition of rare words for ASR systems

Reference Snippet	Baseline Output	Contextualization Output	Metadata Words (truncated)
its very intuitive so when you look at **PyTorch** itself	its very intuitive so when you look at **pie towards** itself	its very intuitive so when you look at **PyTorch** itself	experiences, novel, PyTorch updates, Facebook, machine, AI, language, research, ...
hey assistant call **dharashivkar**	hey assistant call **dharashakar**	hey portal call sachin **dharashivkar**	dharashivkar, john, kelly tom, singh, ...

Utilizing Utterance Specific Context: Biasing Module

- Train RNN-T model using: **Utterance specific context words list** along with **Audio**, **True Transcript**
- We want to let model know what could be next possible output units if the next word produced in transcript was from context words list.
- **Biasing Module:** Built using Trie (data structure) of all utterance specific context words. Trie is used to find what words from context list can be finished from last unfinished word in text history so far
- **Text history so far:** Movies with An
- Queries:
  - Last unfinished word suffix: Android, Anna (Jain et al., "Contextual RNN-T For Open Domain ASR")



Triie with Context Words: Android, Anna, Pytorch

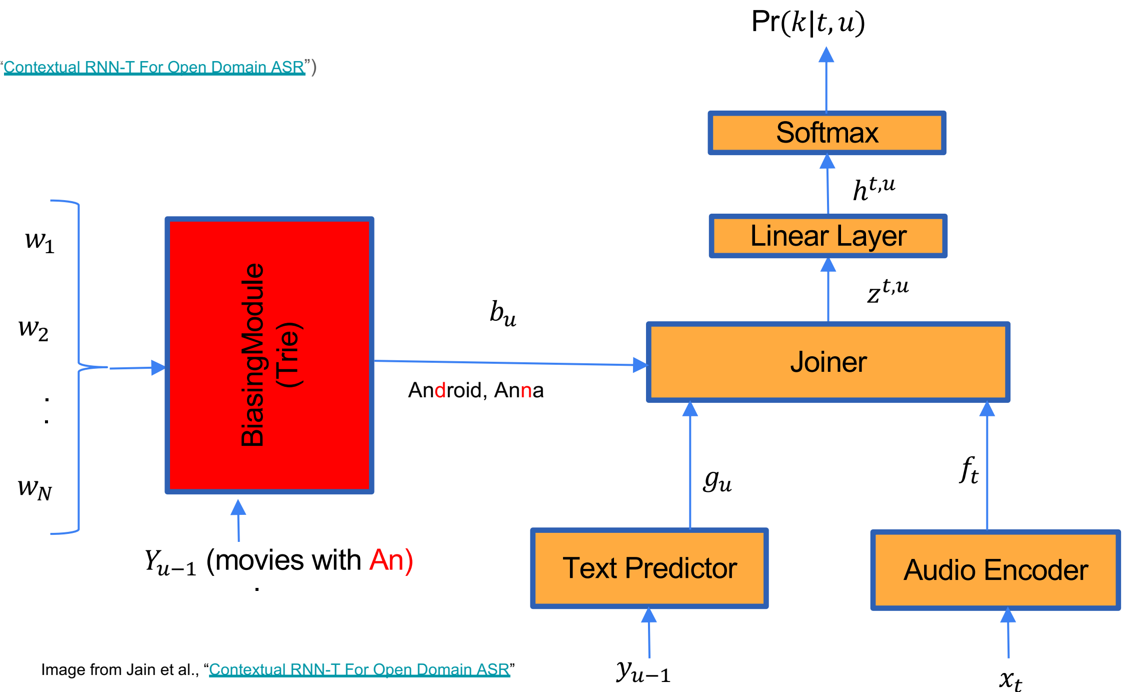


Image from Jain et al., "Contextual RNN-T For Open Domain ASR"

Utilizing Utterance Specific Context: Attention Module

- Train RNN-T model using: **Utterance specific embeddings** (i.e. embeddings of context words: Android, Anna, Pytorch) along with **Audio**, **True Transcript** (Jain et al., "[Contextual RNN-T For Open Domain ASR]")
- **Embedding Extractor:** Extracts embeddings of relevant context information. Here we are using embeddings of context words but we could use other embeddings such as visual embeddings.
- **Attention Module:** Attends between contextual embeddings and output of RNN-T Text Predictor

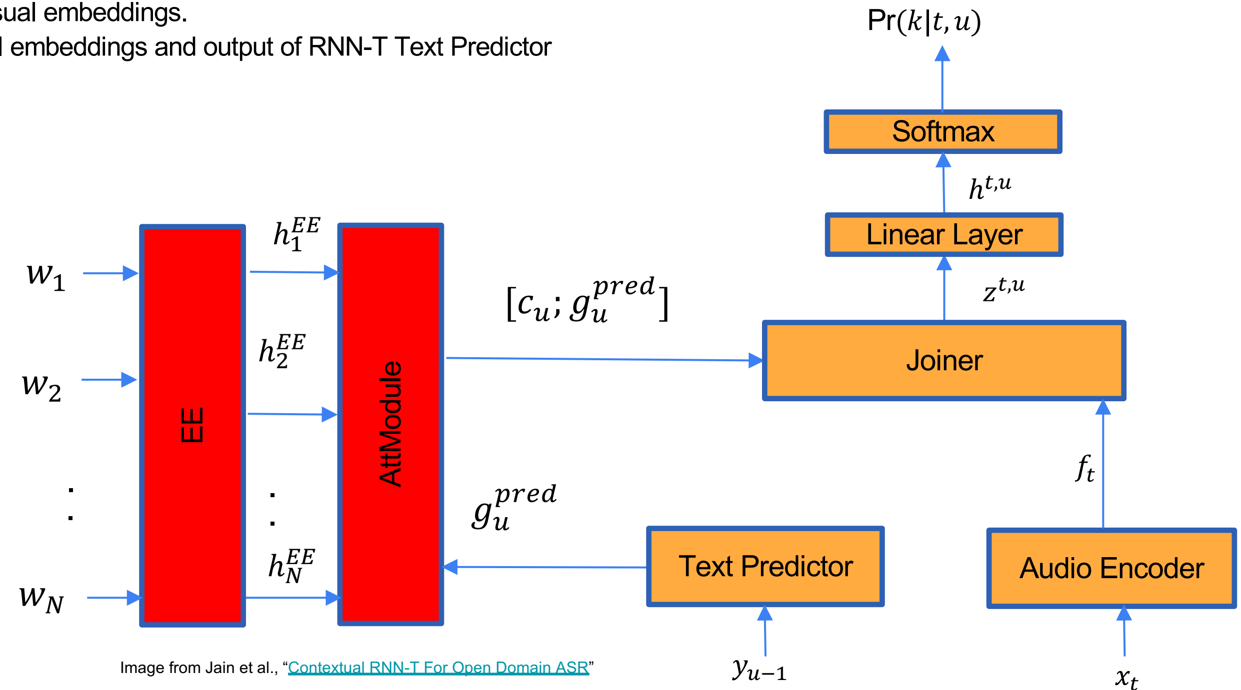


Image from Jain et al., "[Contextual RNN-T For Open Domain ASR]"

What Does Attention Module Learn?

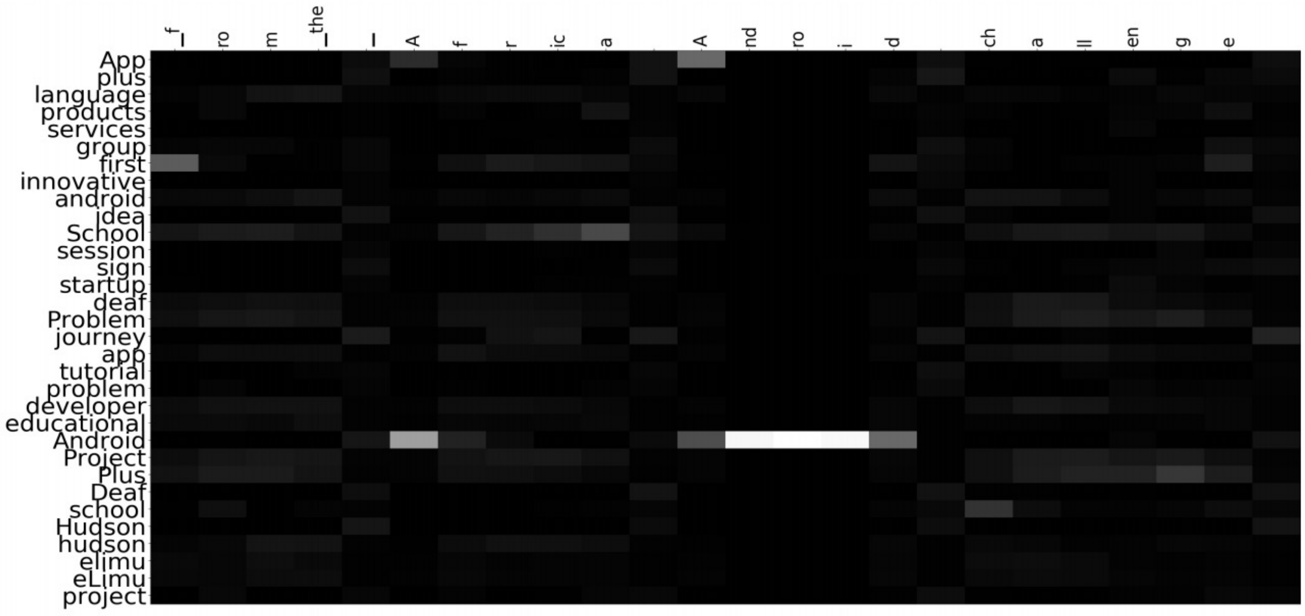
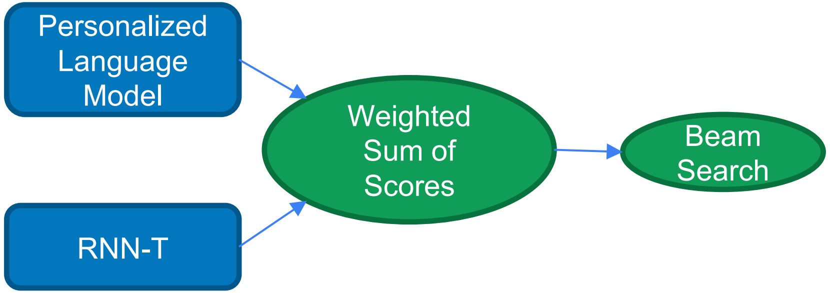


Figure 3: *Visualizing attention weights, $\alpha_{u,n}$ from Equation* (5c), *for the example in Table 1, row 1. The x-axis shows the target units from the hypothesis output by the Contextual RNN-T Model, and the y-axis shows the contextual metadata words (w_i). Darker colors represent values close to zero while brighter colors represent values closer to 1.*

Image from Jain et al., “[Contextual RNN-T For Open Domain ASR]”

Utilizing Utterance Specific Context: Shallow Fusion



- Build a personalized Language Model (LM) using WFST
  - Patterns:
    - Hey Assistant call **@entity**
      - Hey Assistant call Nayan
  - In the wild messaging :
    - … **@name** …
      - message Nayan that I will be late…

- Language Model (LM) has same output units as RNN-T's one.

- **Shallow Fusion** is performed by computing a weighted sum of scores from Language Model and RNN-T model.

- Compute weighted sum of scores from RNN-T(Pr$(k|t, y^*)$) and personalized Language Model (Pr_LM$(k|y^*)$) and use it in beam search.

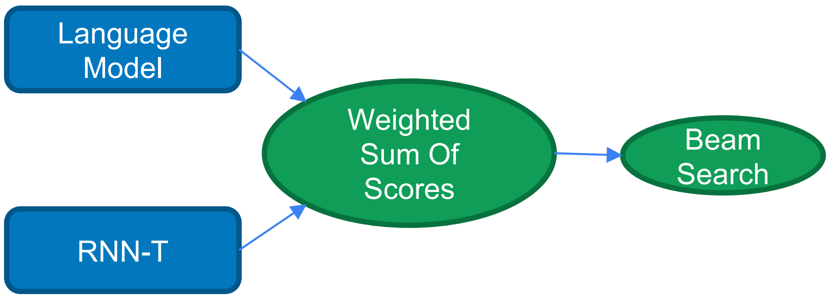
- Boost occurrence of contextual entity (**@entity)** using Personalized Language Model (PLM)

Chen et al., "End-to-end contextual speech recognition using class language models and a token passing decoder" and Le et al., "Contextualized Streaming End-to-End Speech Recognition with Trie-Based Deep Biasing and Shallow Fusion"

Contextualization: Utilizing Large Text Only data with Shallow Fusion

- Language models (LM) built with external text only data allows to model knowledge of the world into ASR system. Helpful when ASR training data is limited. Language model can estimate for example, **Pr_LM("m"| ["P", "r", "e", "s", "I", "d", "e", "n", t", " ", "B", "a", "r", "a", "c", "k", " ","O", "b", "a"])** from text LM only data.

- Compute weighted sum of scores from RNN-T(Pr$(k|t, y^*)$) and Language Model (Pr_LM$(k|y^*)$) and use it in beam search



Choice Of Output Units For RNN-T ASR

- RNN-T ASR produces probability distribution over output units.
- Letters do not take co-occurrence into account.
- Words
  - Large number of output units.
  - Out of vocabulary concern for unseen words in ASR training data.
- Sentence Pieces
  - Configurable number of output units.
  - Combines co-occurring characters in single unit.
  - Example:
    - Hello World encoded as [[Hello] [_Wor] [ld]]
  - Number of output units larger than letters but less number of runs for text predictor network.
    - If a phrase is of 11 letters but could be re-presented by 3 sentence pieces then we only need to run the text predictor network ~1/4th time.

Optimization: Beam Search Decoding

- Limit number of expanded hypothesizes that are added in A

Expand best hypo from A only with the output symbols that are within ***expand_beam*** value of best expansion output symbol

- Additional Conditions for exiting beam search at audio frame *t*

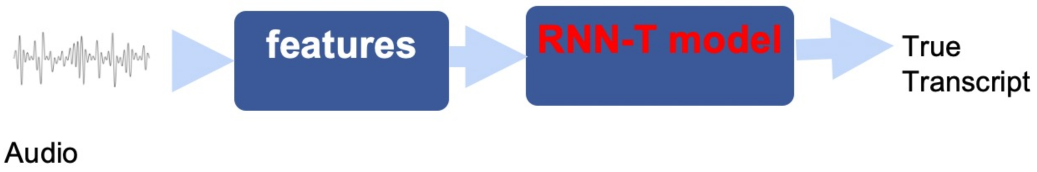
Exit if the best hypothesis in B is better by more than ***state_beam*** compared to best hypothesis in A

Algorithm 1 Improved RNNT Beam Search

```
Initialize: B = {∅}; Pr(∅) = 1
for t = 1 to T do
  A = B
  B = {}
  for y in A do
    Pr(y) += Σ_{ŷ ∈ pref(y) ∩ A} Pr(ŷ) Pr(y | ŷ, t)
  end for
  while B contains less than W elements more probable than the most probable in A do
    y* = most probable in A
    a_best_prob = max probability in A
    b_best_prob = max probability in B
    if log(b_best_prob) ≥ state_beam + log(a_best_prob) then
      break {meet beam search and break while loop}
    end if
    Remove y* from A
    Pr(y*) = Pr(y*) Pr(∅ | y, t)
    Add y* to B
    best_prob = max_{k ∈ non_blank} Pr(k | y*, t)
    for k ∈ Y do
      if log(Pr(k | y*, t)) ≥ log(best_prob) − expand_beam then
        Pr(y* + K) = Pr(y*) Pr(k | y*, t)
        Add y* + k to A
      end if
    end for
  end while
  Remove all but the W most probable from B
end for
return y with highest log Pr(y)/|y| in B
```

Jain et al., "RNN-T For Latency Controlled ASR With Imoroved Beam Search"

More Details On Training RNN-T model



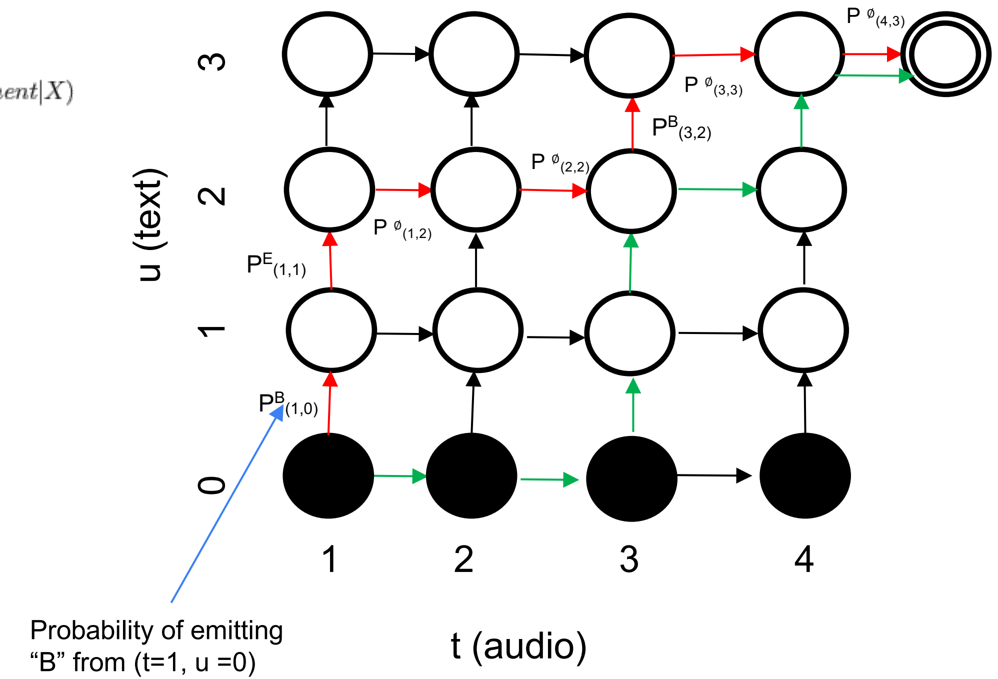
Training: Lattice With Complete Set Of Alignments

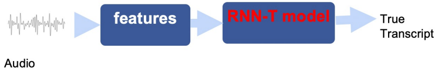
$$P(BEE|X) = \sum_{alignment} P(alignment, BEE|X)$$

$$\sum_{alignment} P(BEE|alignment, X)P(alignment|X)$$

$$P(BEE|alignment, X) = 1 \qquad \sum_{alignment} P(alignment|X)$$

- A naïve implementation is computationally heavy: Use dynamic programming to efficiently compute this.





Graves et al., "[Sequence transduction with recurrent neural networks](#)"

Training: Forward Variable (alpha)

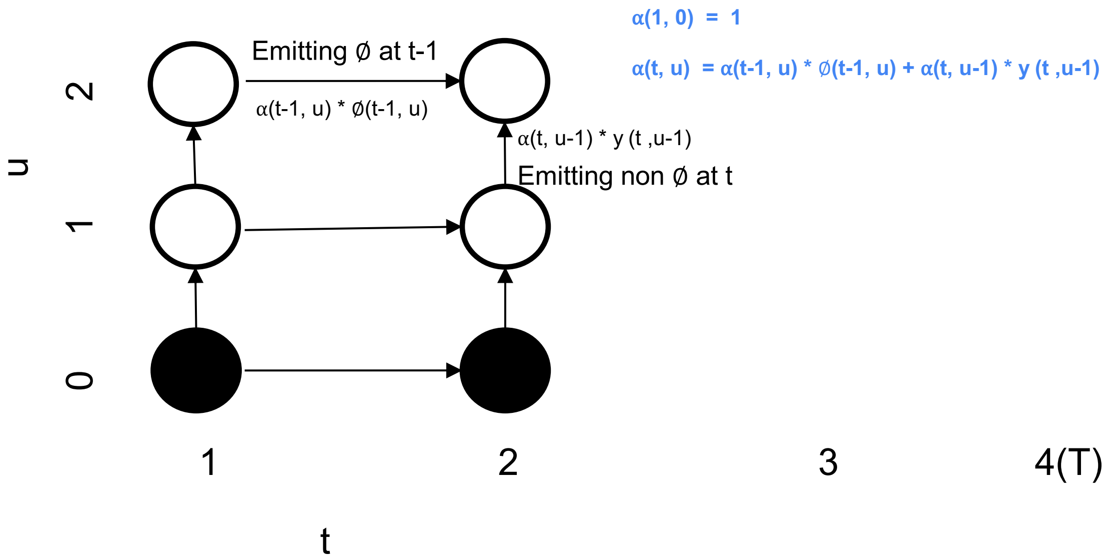
$$y(t, u) \equiv \Pr(y_{u+1} | t, u)$$
$$\varnothing(t, u) \equiv \Pr(\varnothing | t, u)$$

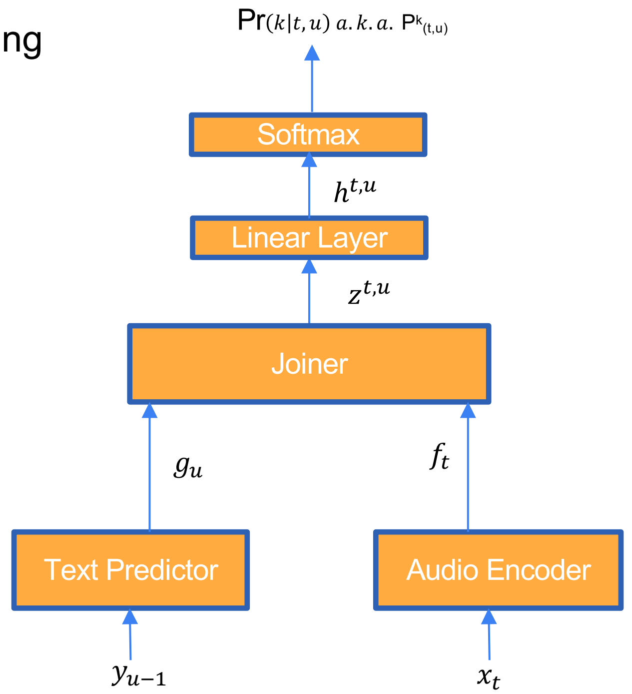
y (t ,u) is probability of emitting next symbol (u + 1) in output sequence from (t, u)

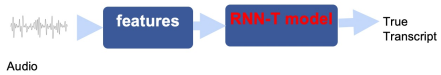
α(t, u) is probability of outputting [: *u*] symbols during [: *t*] frames

α(1, 0) = 1

α(t, u) = α(t-1, u) * ∅(t-1, u) + α(t, u-1) * y (t ,u-1)





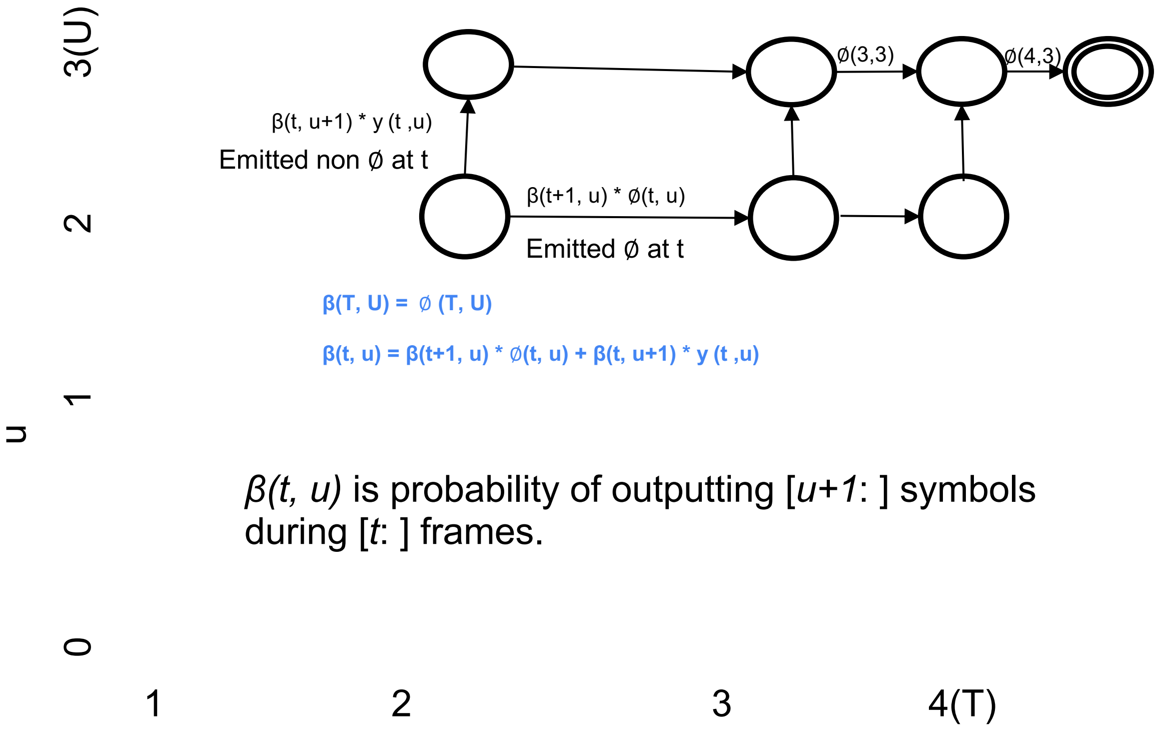


Graves et al., "Sequence transduction with recurrent neural networks"

Training: Backward Variable (beta)

$$y(t,u) \equiv \Pr(y_{u+1}|t,u)$$
$$\varnothing(t,u) \equiv \Pr(\varnothing|t,u)$$

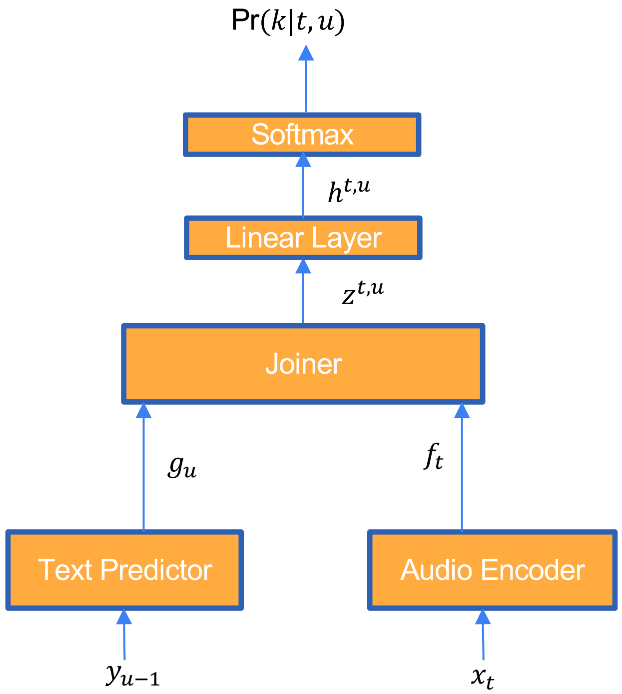
y (t ,u) is probability of emitting next symbol (u + 1) in output sequence from (t, u)

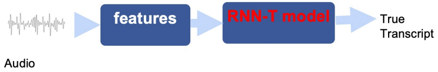


β(T, U) = ø (T, U)

β(t, u) = β(t+1, u) * ø(t, u) + β(t, u+1) * y (t ,u)

β(t, u) is probability of outputting [*u+1*:] symbols during [*t*:] frames.



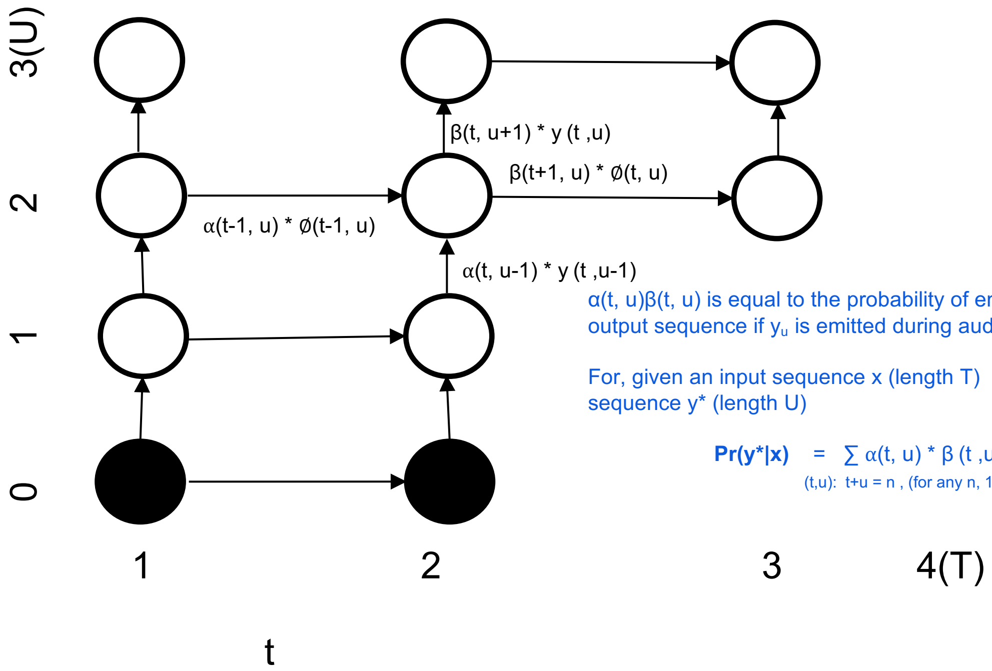


Graves et al., "Sequence transduction with recurrent neural networks"

Training: α(t, u) * β(t, u)

$$y(t,u) \equiv \Pr(y_{u+1}|t,u)$$
$$\varnothing(t,u) \equiv \Pr(\varnothing|t,u)$$

y (t ,u) is probability of emitting next symbol (u + 1) in output sequence from (t, u)

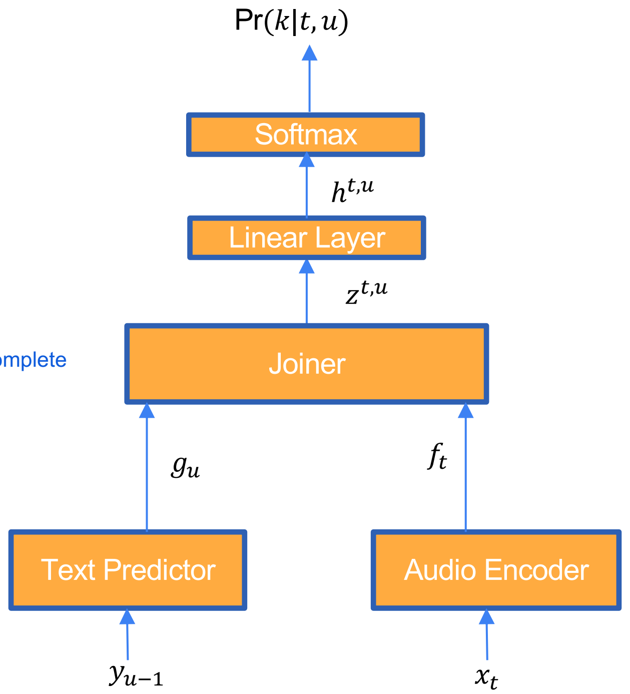


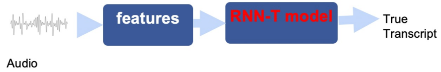
α(t, u)β(t, u) is equal to the probability of emitting the complete output sequence if y_u is emitted during audio frame *t.*

For, given an input sequence x (length T) and a target sequence y* (length U)

Pr(y*|x) = ∑ α(t, u) * β (t ,u)

(t,u): t+u = n , (for any n, 1 ≤ n ≤ U + T)





Graves et al., "Sequence transduction with recurrent neural networks"

Training: Gradient Descent

$$y(t,u) \equiv \Pr(y_{u+1}|t,u)$$
$$\varnothing(t,u) \equiv \Pr(\varnothing|t,u)$$

y (t ,u) is probability of emitting next symbol (u + 1) in output sequence from (t, u)

For Target sequence y^*

$$\mathcal{L} = -\ln \Pr(\boldsymbol{y}^*|\boldsymbol{x})$$

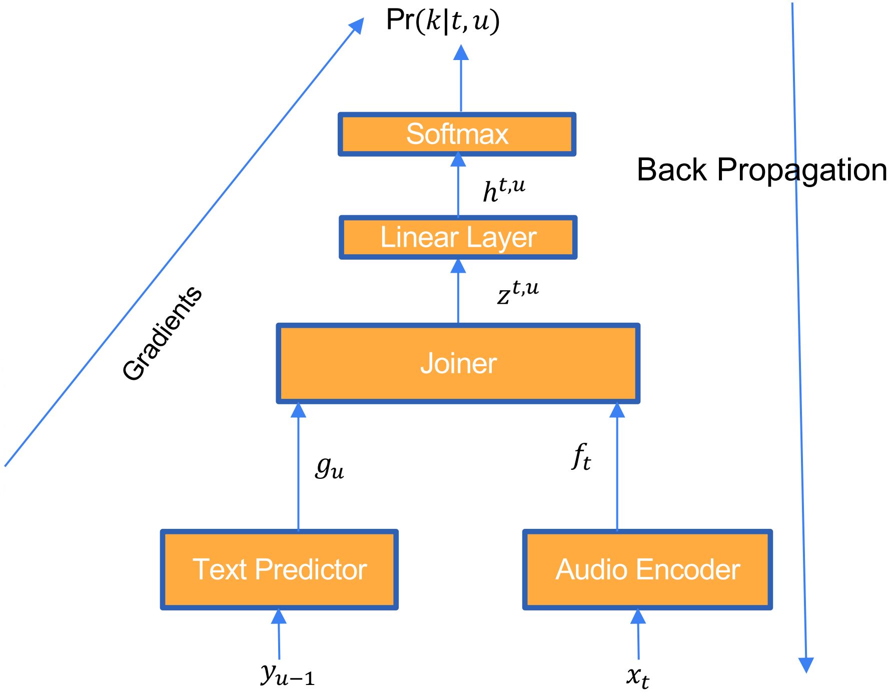
α(t, u) = α(t-1, u) * ∅(t-1, u) + α(t, u-1) * y (t ,u-1)

β(t, u) = β(t+1, u) * ∅(t, u) + β(t, u+1) * y (t ,u)

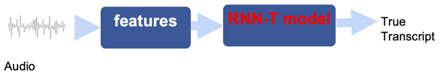
Pr(y*|x) = ∑ α(t, u) * β (t ,u)
(t,u): t+u = n (for any n, 1 ≤ n ≤ U + T)

$$\frac{\partial \mathcal{L}}{\partial \Pr(k|t,u)} = -\frac{\alpha(t,u)}{\Pr(\boldsymbol{y}^*|\boldsymbol{x})} \begin{cases} \beta(t,u+1) \text{ if } k = y_{u+1} \\ \beta(t+1,u) \text{ if } k = \varnothing \\ 0 \text{ otherwise} \end{cases}$$

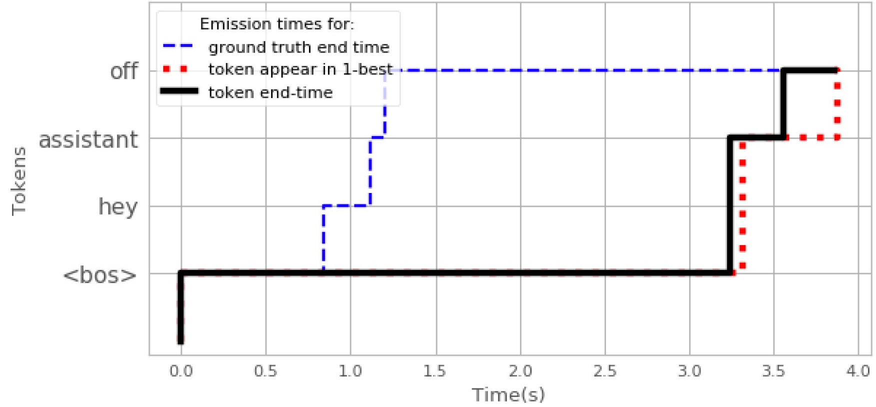
Image from Graves et al., "Sequence transduction with recurrent neural networks"



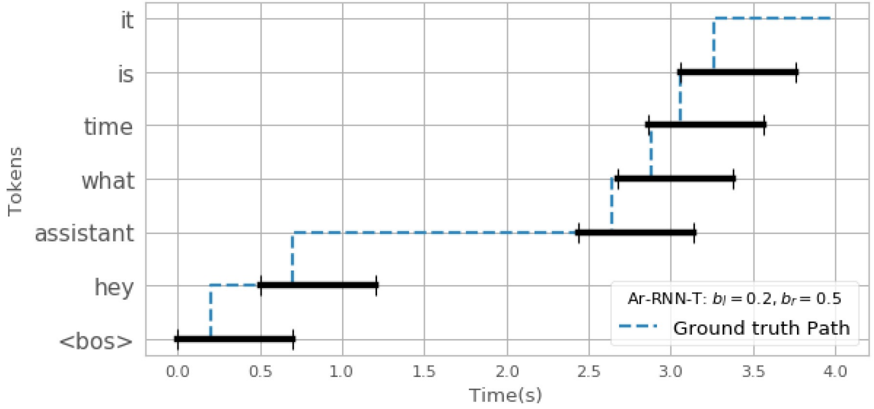
Graves et al., "Sequence transduction with recurrent neural networks"



Improving Training Optimization And Emission Delays



Token Emission Delay Problem: During inference, tokens are emitted with considerable delay after they are spoken



Restrict alignment paths: Helps in token emission delay and improving training speed.

Mahadeokar et al. "[Alignment Restricted Streaming Recurrent Neural Network Transducer]"

Further Reading

- CTC: Graves et al., Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks
- RNN-T: Graves et al., Sequence Transduction with Recurrent Neural Networks
- On Device RNN-T ASR:
  - He et al., Streaming end-to-end speech recognition for mobile devices
  - Yuan et al., Optimizing speech recognition for the edge
- Improving RNN-T beam search
  - Jain et al., RNN-T For Latency Controlled ASR With Imoroved Beam Search
- Contextualization:
  - Jain et al., Contextual RNN-T For Open Domain ASR
  - Le et al., "Contextualized Streaming End-to-End Speech Recognition with Trie-Based Deep Biasing and Shallow Fusion"
- RNN-T Variants:
  - Variani et al., Hybrid Autoregressive Transducer (HAT)
  - Tripathi et al., Monotonic RNN-T
- Comparing RNN-T with other ASR techniques
  - Zhang et al., Benchmarking LF-MMI, CTC and RNN-T Criteria for Streaming ASR
  - Jain et al., RNN-T For Latency Controlled ASR With Imoroved Beam Search
- Sentence Piece
  - Kudo et al., A simple and language independent subword tokenizer and detokenizer for Neural Text Processing
- Good read on tries: https://medium.com/basecs/trying-to-understand-tries-3ec6bede0014 (Vaidehi Joshi)

Thanks

- Many thanks to Rohit Prabhavalkar, Mark Tygert, Michael Picheny, Nayan Singhal, Kritika Singh, Xiaohui Zhang, Prady Prakash, Duc Le, Yuan Shangguan, Paco Guzmán, Yatharth Saraf and Mike Seltzer for brainstorming with ideas to improve content of this presentation.