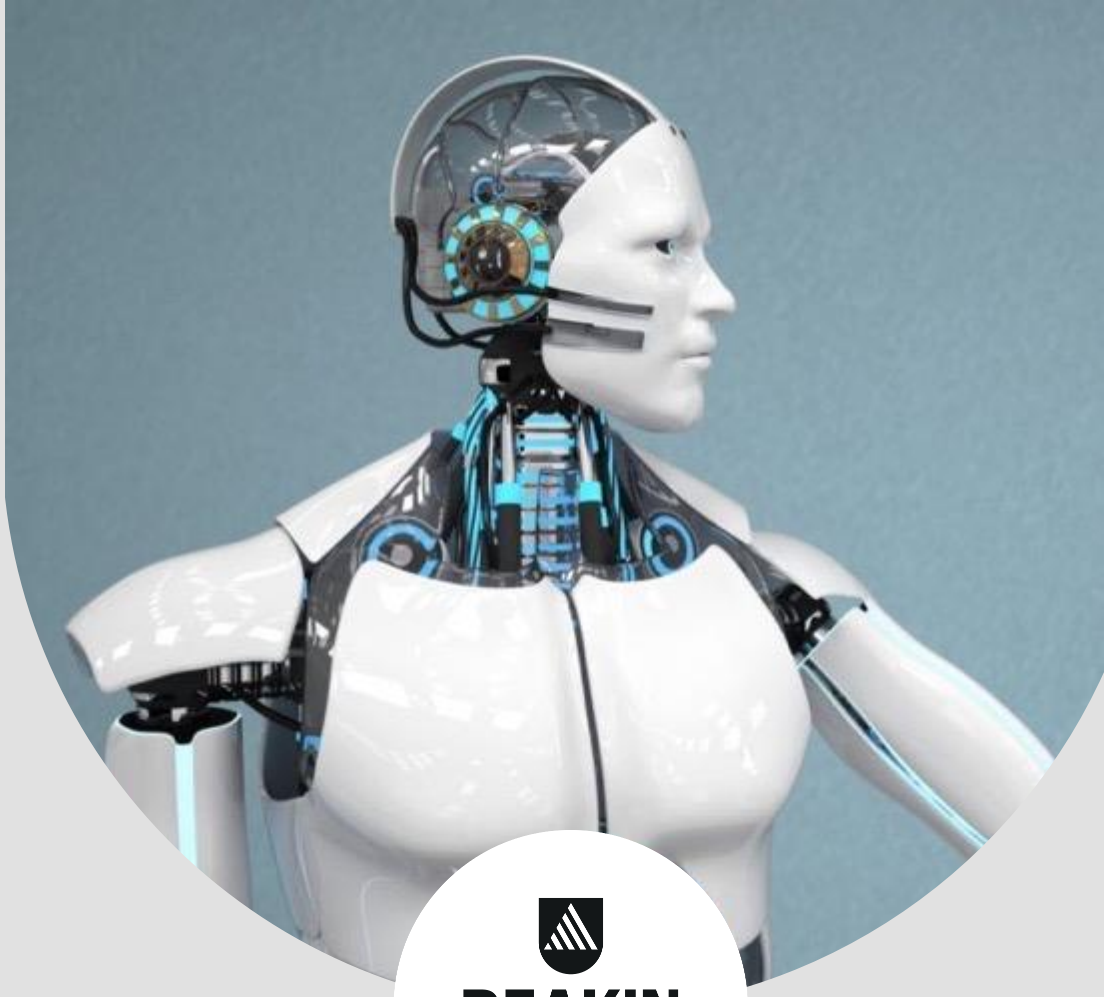
SIT330-770: Natural Language Processing

Week 9 – Speech Processing & ASR

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

reda.bouadjenek@deakin.edu.au



DEAKIN UNIVERSITY

SIT330-770: Natural Language Processing

Week 9. 1 - Introduction to Automatic Speech Recognition

Dr. Mohamed Reda Bouadjenek

School of Information Technology,
Faculty of Sci Eng & Built Env

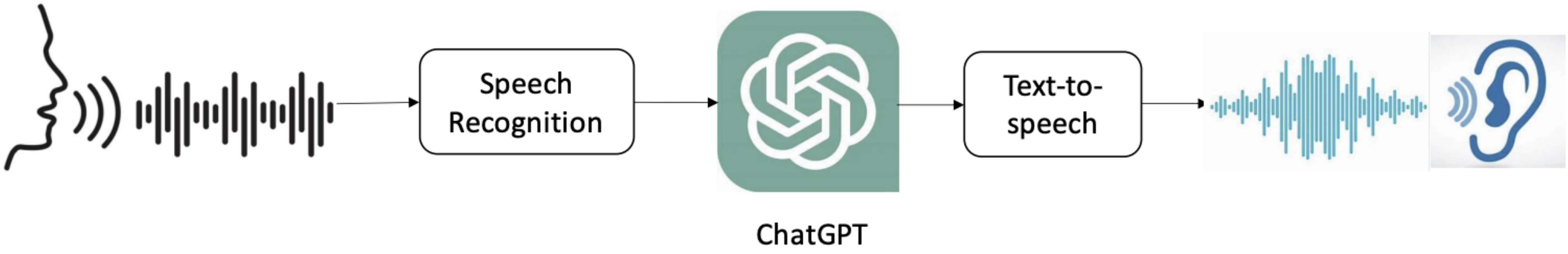
What is speech recognition?

- Speech-to-text transcription
  - Transform recorded audio into a sequence of words
  - Just the words, no meaning…. But do need to deal with acoustic ambiguity: “Recognise speech?” or “Wreck a nice beach?”
  - Speaker diarization: Who spoke when?
  - Speech recognition: what did they say?
  - Paralinguistic aspects: how did they say it? (timing, intonation, voice quality)
  - Speech understanding: what does it mean?

Applications of ASR

- Dictation
- Language learning
- Smart speakers (Alexa, Siri)
- Accessibility for hearing impaired
- Voice command
- Automatic captioning
- Audio indexing
- Machine translation
- Meeting understanding and summarization
- Call center analysis
- TV remote
- ...

Enable ChatGPT with voice input/output



Why is speech recognition difficult?

- Several sources of variation
  - **Size**
    - Number of word types in vocabulary, perplexity
  - **Speaker**
    - Tuned for a particular speaker, or speaker-independent? Adaptation to speaker characteristics
  - **Acoustic environment**
    - Noise, competing speakers, channel conditions (microphone, phone line, room acoustics)
  - **Style**
    - Continuously spoken or isolated? Planned monologue or spontaneous conversation?
  - **Accent/dialect**
    - Recognise the speech of all speakers who speak a particular language
  - **Language spoken**
    - There are many languages beyond English, Mandarin Chinese, Spanish, . . . What is the difference between a dialect and a language?

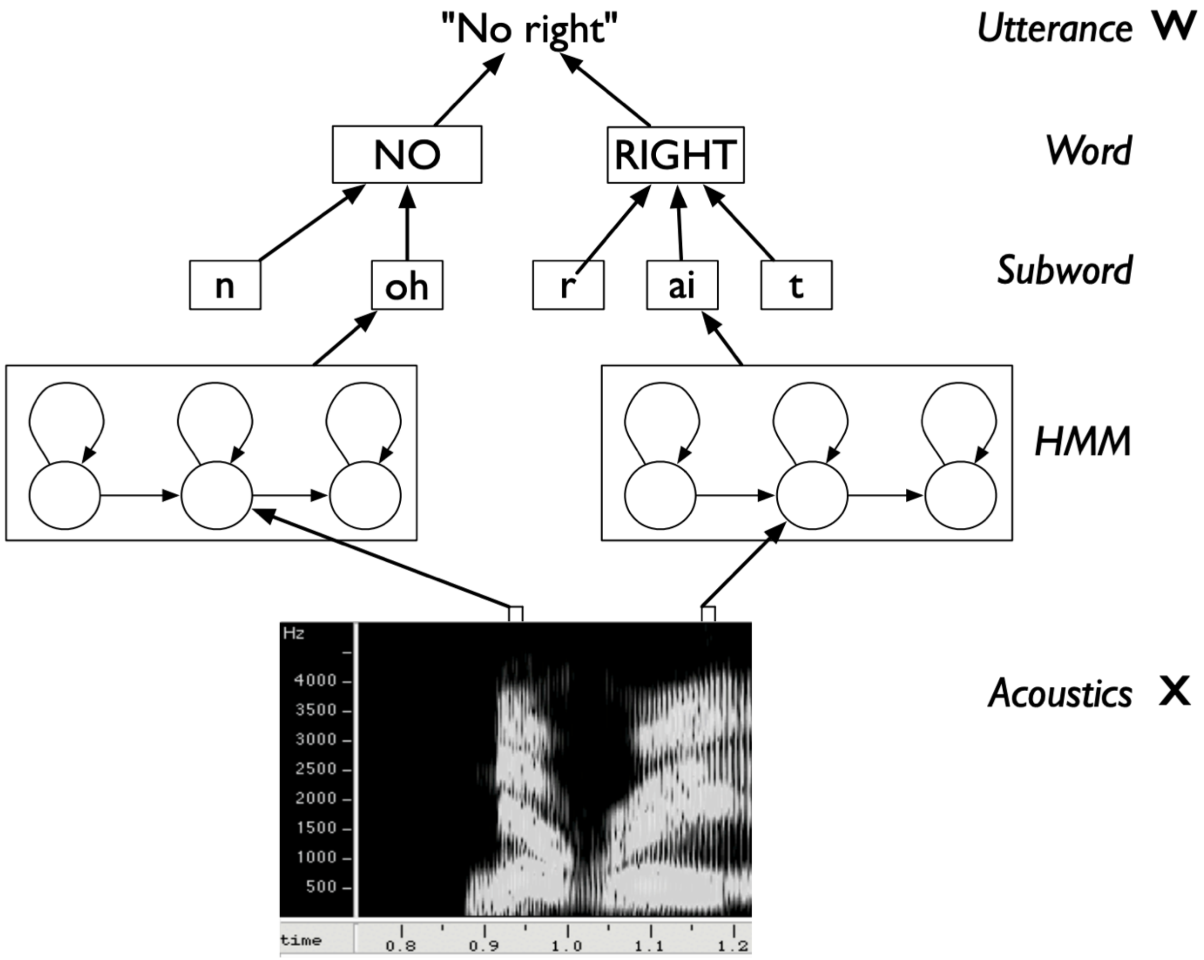
SIT330-770: Natural Language Processing

Week 9. 2 - Statistical modeling for Automatic Speech Recognition

Dr. Mohamed Reda Bouadjenek

School of Information Technology,
Faculty of Sci Eng & Built Env

Hierarchical modelling of speech



Statistical Speech Recognition

- If **X** is the sequence of acoustic feature vectors (observations) and **W** denotes a word sequence, the most likely word sequence **W*** is given by

$$W^* = \underset{w}{\operatorname{argmax}}\, P(W|X)$$

- Applying Bayes' Theorem:

$$\begin{aligned} P(W|X) &= \frac{P(X|W)P(W)}{P(X)} \\ &\propto P(X|W)P(W) \\ W^* &= \underset{w}{\operatorname{argmax}}\, \boxed{P(X|W)}\ \boxed{P(W)} \end{aligned}$$

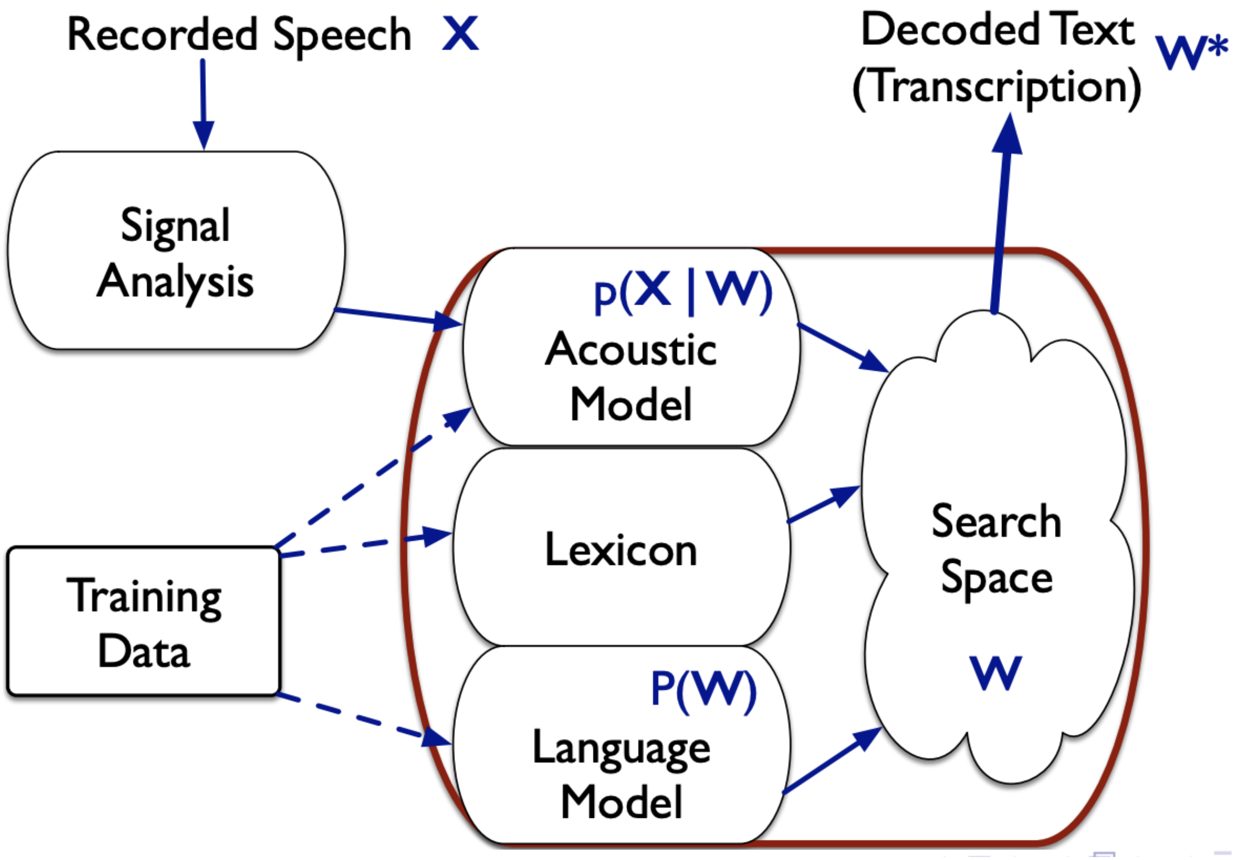
Language model

Acoustic model

Speech Recognition Components

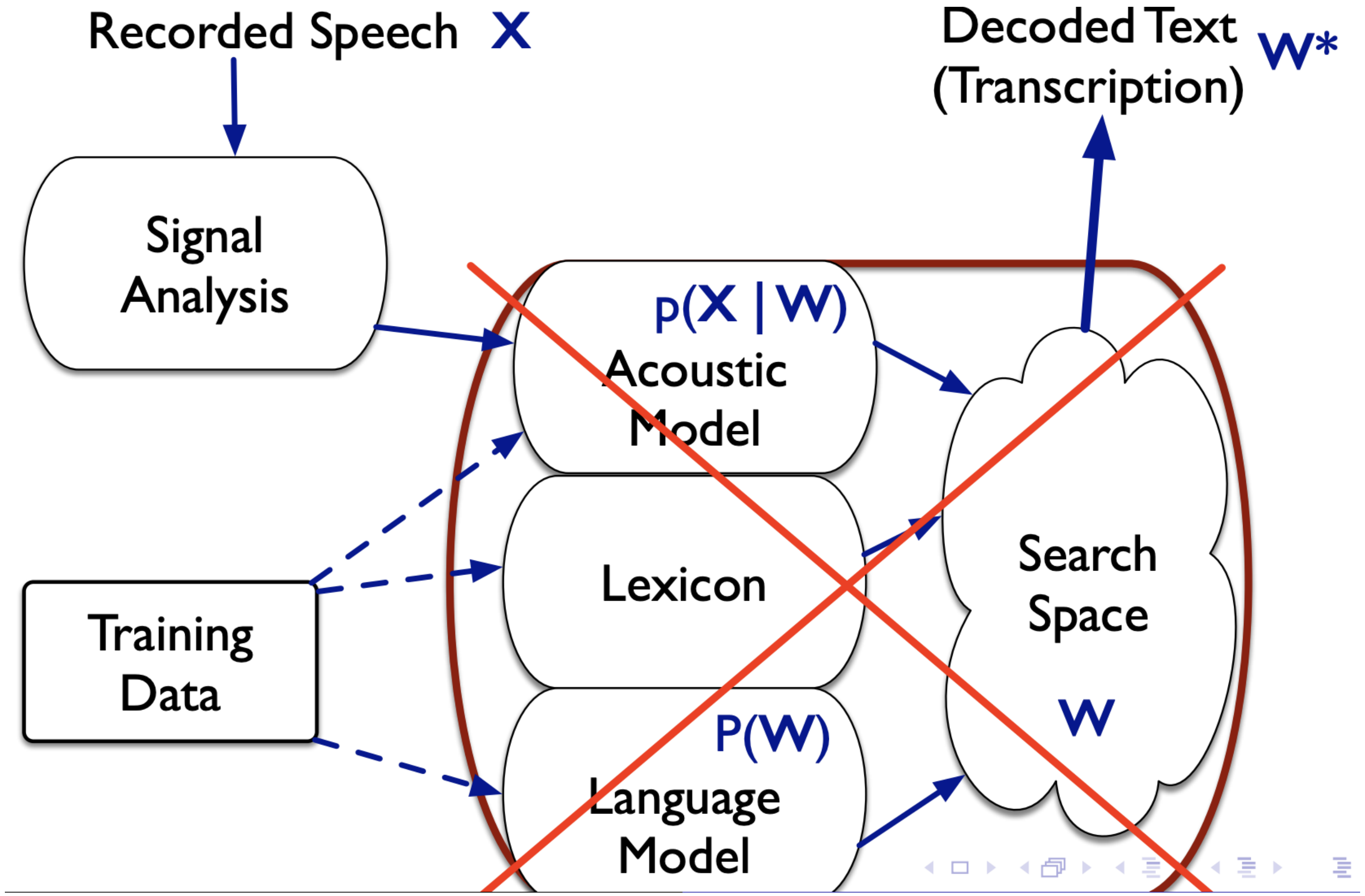
$$W^* = \underset{w}{\operatorname{argmax}}\, P(W|X)$$

- Use an acoustic model, language model, and lexicon to obtain the most probable word sequence **W*** given the observed acoustics **X**



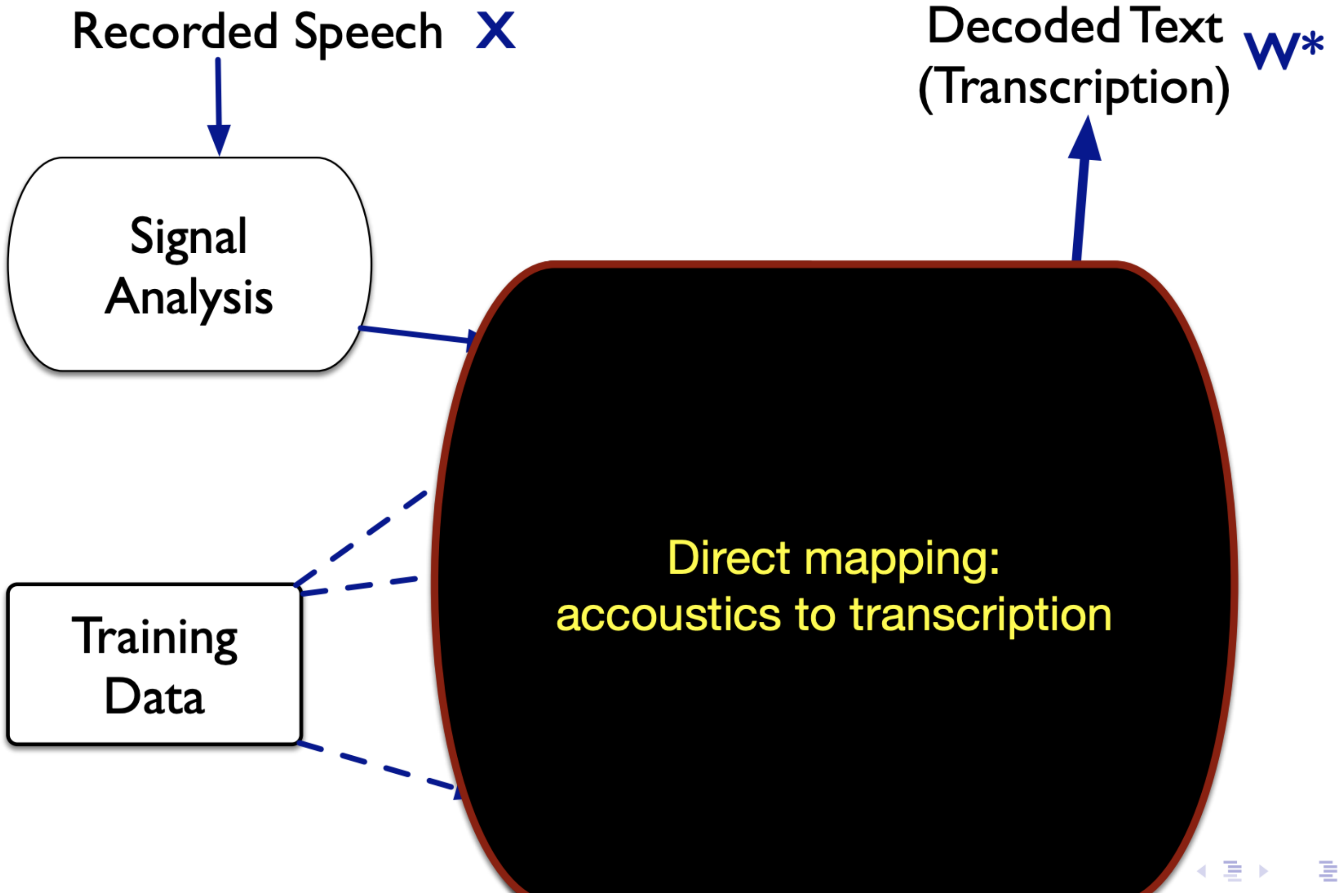
Alternative approach: End-to-end systems

- Directly model transforming an input acoustic sequence into an output word or character sequence



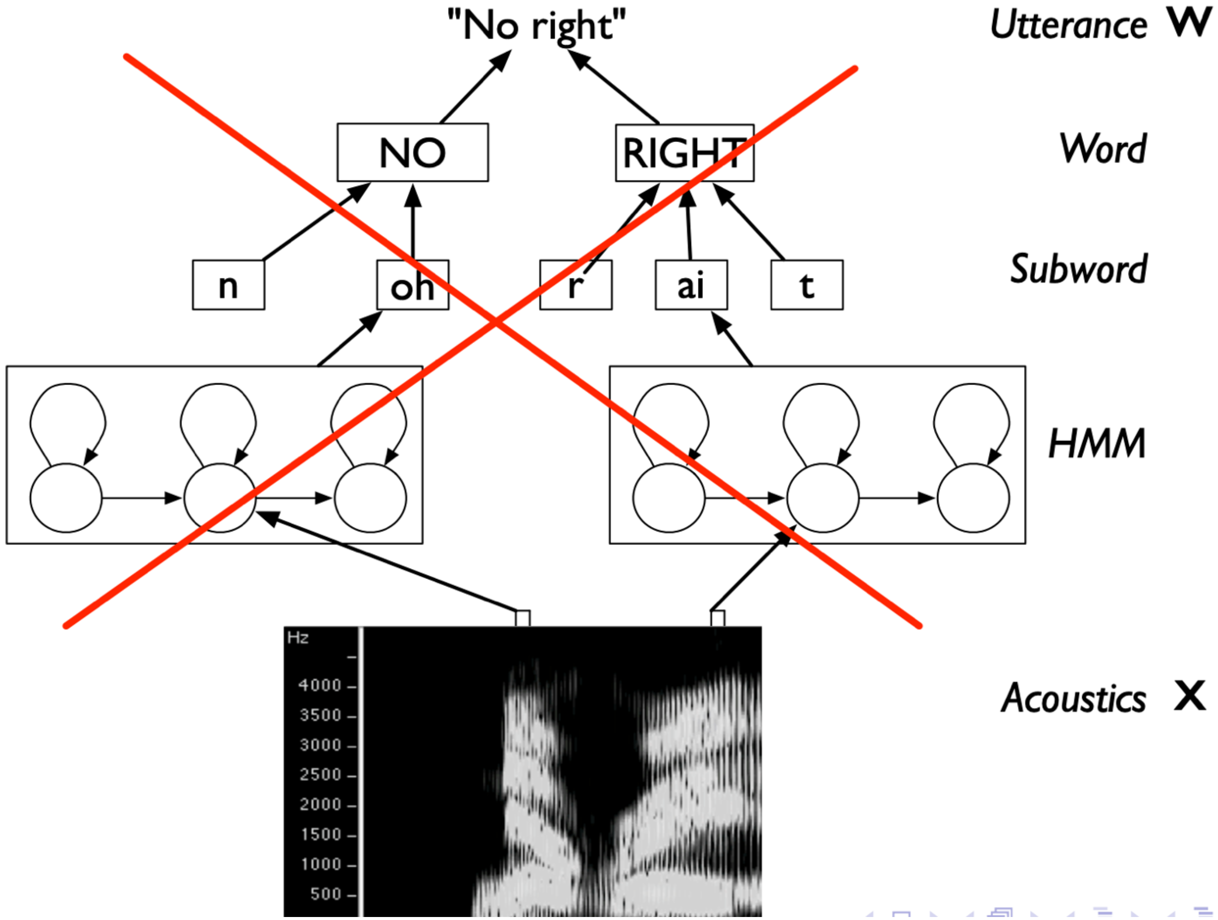
Alternative approach: End-to-end systems

- Directly model transforming an input acoustic sequence into an output word or character sequence



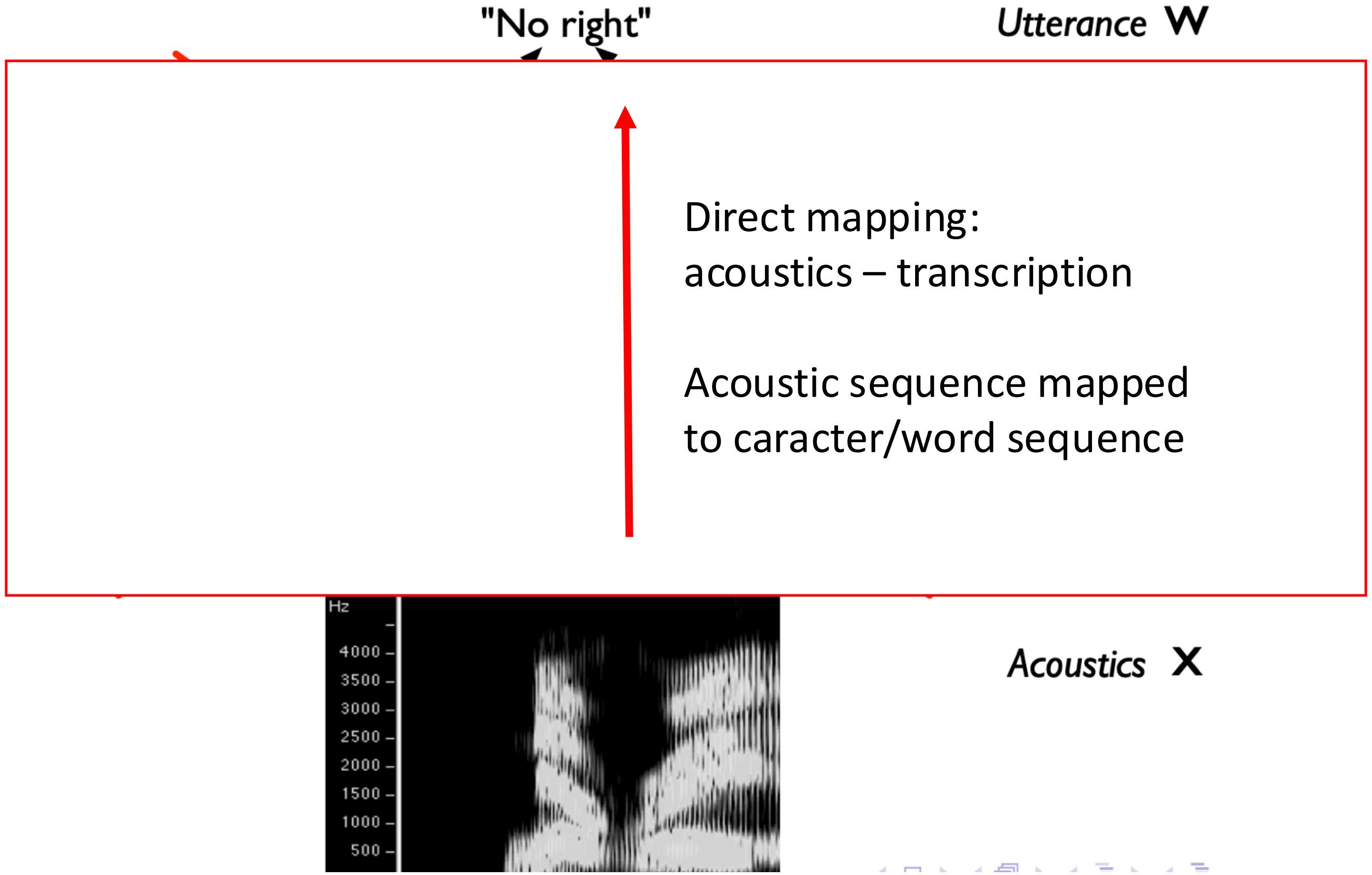
Alternative approach: End-to-end systems

- Directly model transforming an input acoustic sequence into an output word or character sequence



Alternative approach: End-to-end systems

- Directly model transforming an input acoustic sequence into an output word or character sequence



SIT330-770: Natural Language Processing

Week 9. 3 - Evaluation Metrics

Dr. Mohamed Reda Bouadjenek

School of Information Technology,
Faculty of Sci Eng & Built Env

Evaluation Metrics

- Reference:
  - The quick brown fox jumped over the lazy dog
- Hypothesis:
  - The quick brown fox **jumps** over ---- lazy dog **too**
- Word error rate:
  - $WER = D+S+I \; N$
    - D: number of deleted words
    - S: number of subsituted words
    - I: number of inserted words
    - N: number of reference words
- Readability: whether the recognized text is easy to read by human.

SIT330-770: Natural Language Processing

Week 9. 4 - Deep learning for ASR

Dr. Mohamed Reda Bouadjenek

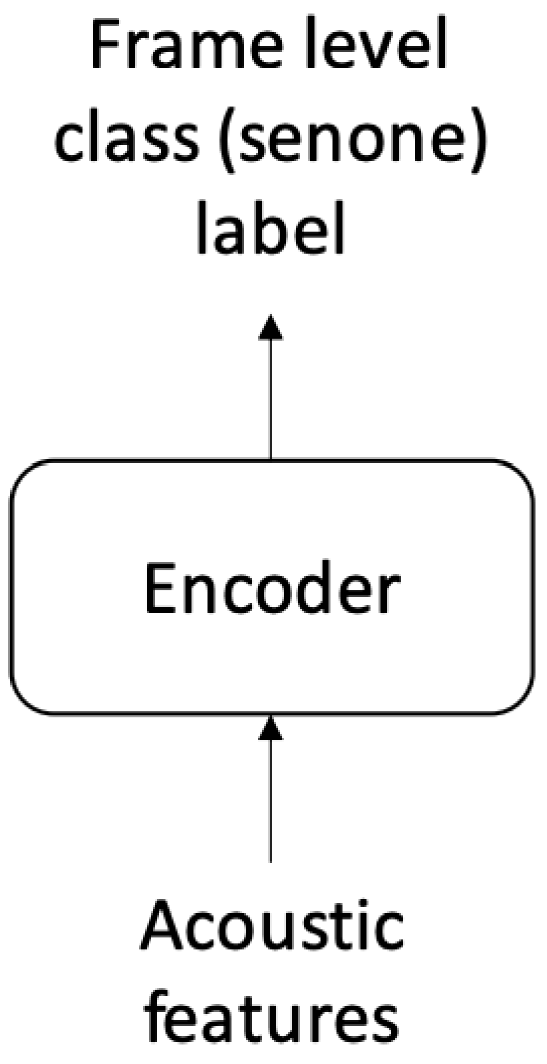
School of Information Technology,
Faculty of Sci Eng & Built Env

Deep learning for ASR

- Hybrid system: only replace HMM/GMM acoustic model with neural networks
- End-to-end ASR: replace the whole ASR system with neural works

Hybrid acoustic model

- Replace the generative HMM/GMM with a discriminative neural networks
- HMM/GMM models $p(o_t|s_t)$
- Hybrid models $p(s_t|o_t)$
- Common practices
  - Train an HMM/GMM first
  - Use it to align the label (senone sequences) to the feature sequence.
  - Train neural networks to predict frame level senone labels



Encoder Structures

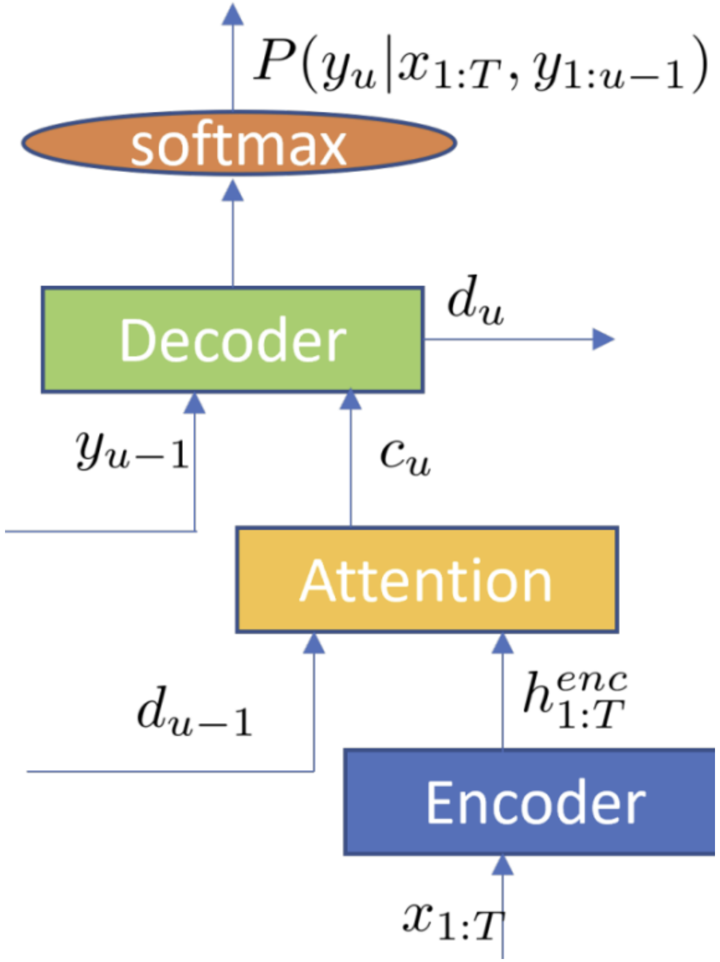
- DNN
- CNN
- LSTM
- Transformer
- Or any combination of them

End-to-end ASR

- End-to-end ASR systems try to do ASR with a single model
- Three main approaches
  - Connectionist Temporal Classification
  - RNN Transducers
  - Sequence-to-Sequence

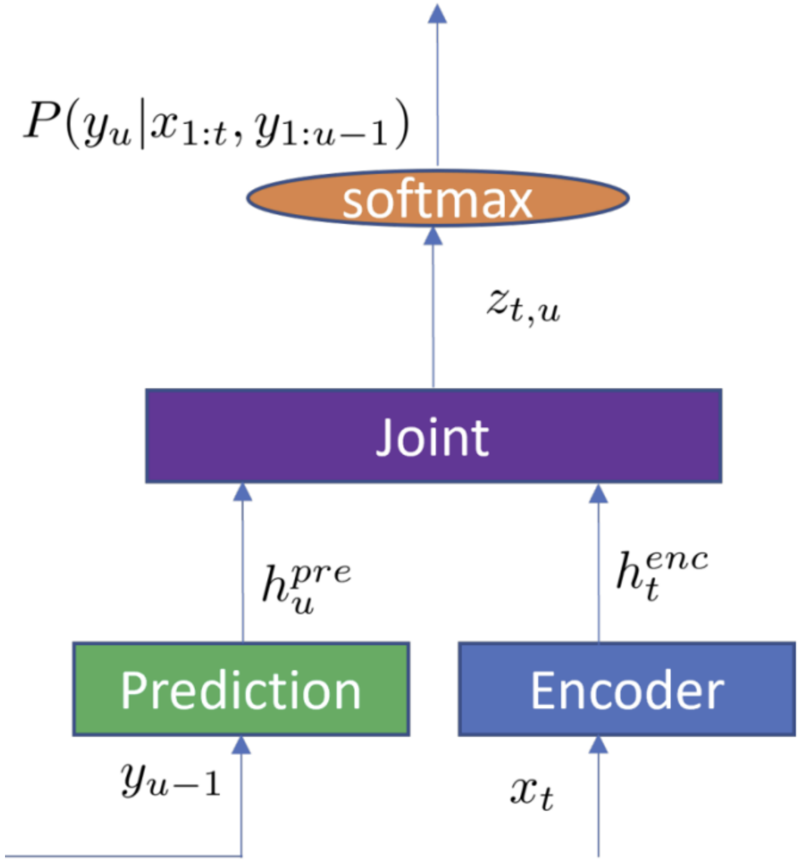
Sequence-to-sequence (S2S)

- S2S is also called attention encoder decoder (AED)
- Encoder: similar to acoustic model
- Attention: alignment model
- Decoder: similar to pronunciation and language model
- Offline model



RNN Transducers (RNN-T)

- Called RNN-T because originally RNN is used as the encoder model structure.
- Newer models uses transformers or conformers as encoder
- A native streaming model



SIT330-770: Natural Language Processing

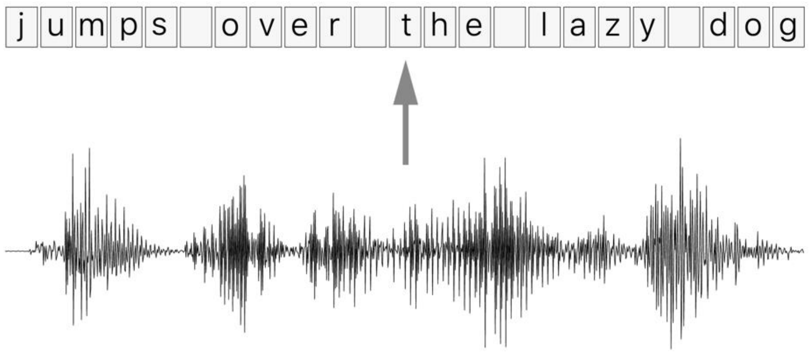
Week 9. 5 - The alignment problem

Dr. Mohamed Reda Bouadjenek

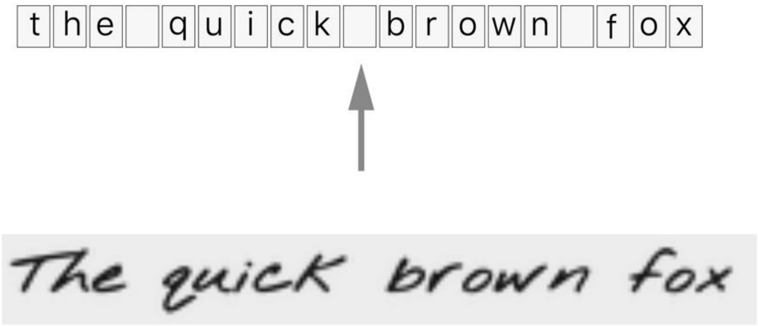
School of Information Technology,
Faculty of Sci Eng & Built Env

The alignment problem

- We have a data set of speech, handwriting, other sequential data and the corresponding transcripts
- **Problem: we don't know how the outputs align to the inputs**
  - **i.e., which frame(s) of the input correspond to which output frame**



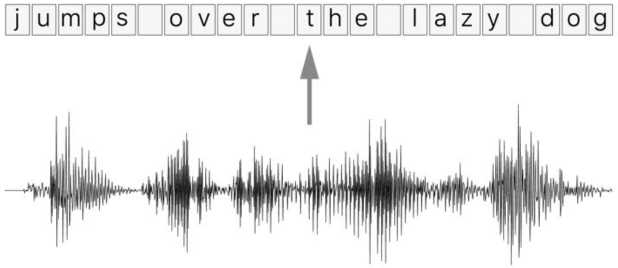
Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.



Handwriting recognition: The input can be (x, y) coordinates of a pen stroke or pixels in an image.

The alignment problem: Naïve solutions

- We could devise a rule like “one character corresponds to ten inputs”.
  - But people’s rates of speech vary, so this type of rule can always be broken.
- Another alternative is to hand-align each character to its location in the audio.
  - May work well, but we’d know the ground truth for each input time-step
  - For any reasonably sized dataset this is prohibitively time consuming.



Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.

Solution: **Connectionist Temporal Classification (CTC) is a way to get around not knowing the alignment between the input and the output**

Problem definition

- Given:
  - A sequence $X=[x_1,x_2,\ldots,x_T]$ (audio)
  - The corresponding output sequence $Y=[y_1,y_2,\ldots,y_U]$ (transcript)
- We want to find an accurate mapping from X to Y
- Challenges:
  - Both X and Y can vary in length
  - The ratio of the lengths of X and Y can vary.
  - We don't have an accurate alignment (correspondence of the elements) of X and Y
- The CTC algorithm overcomes these challenges and for a given X it gives an output distribution over all possible Y
  - We can use this distribution either to *infer* a likely output or to assess the *probability* of a given output.

SIT330-770: Natural Language Processing

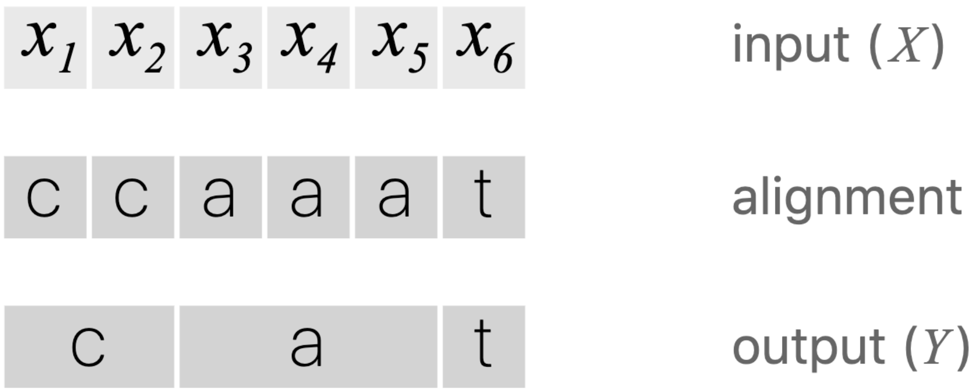
Week 9. 6 -Automatic Speech Recognition using Connectionist Temporal Classification

Dr. Mohamed Reda Bouadjenek

School of Information Technology,
Faculty of Sci Eng & Built Env

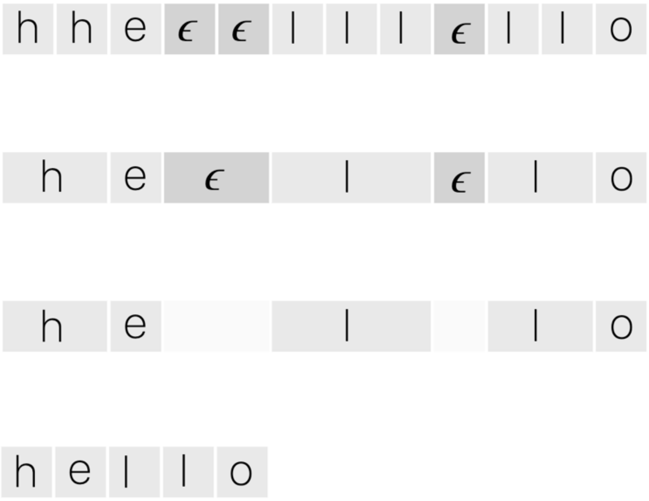
The algorithm: Alignment

- Assume the input has length six and Y = [c, a, t]. One way to align X and Y is to assign an output character to each input step and collapse repeats
- This approach has two problems:
  - It doesn't make sense to force every input step to align to some output
  - We have no way to produce outputs with multiple characters in a row.
    - The alignment **[h, h, e, l, l, l, o]** collapses to "**helo**"



The algorithm: CTC Alignment

- CTC introduces a new token **ϵ** called the blank token
- The **ϵ** token doesn't correspond to anything
- We allow any alignment which maps to Y after merging repeats and removing **ϵ** tokens:



First, merge repeat characters.

Then, remove any ϵ tokens.

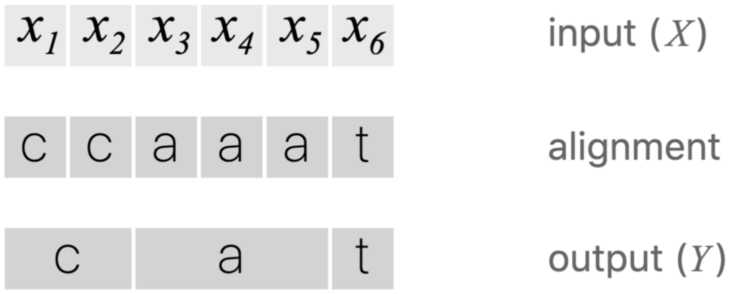
The remaining characters are the output.

The algorithm: CTC Alignment Examples

Valid Alignments	Invalid Alignments	
ϵ c c ϵ a t	c ϵ c ϵ a t	corresponds to $Y =$ [c, c, a, t]
c c a a t t	c c a a t	has length 5
c a ϵ ϵ ϵ t	c ϵ ϵ ϵ t t	missing the 'a'

The algorithm: CTC Alignment Properties

- The allowed alignments between X and Y are monotonic.
  - If we advance to the next input, we can keep the corresponding output the same or advance to the next one.
- The alignment of X to Y is many-to-one.
  - One or more input elements can align to a single output element but not vice-versa.
- The length of Y cannot be greater than the length of X.



The algorithm: Loss Function

- The CTC alignments gives us a probability of an output sequence
- The CTC objective for a single (X,Y) pair is:

$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC conditional **probability**

marginalizes over the set of valid alignments

computing the **probability** for a single alignment step-by-step.

We start with an input sequence, like a spectrogram of audio.

The input is fed into an RNN, for example.

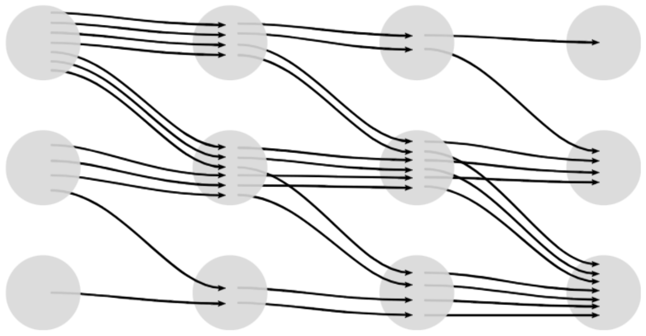
The network gives $p_t(a \mid X)$, a distribution over the outputs {h, e, l, o, ϵ} for each input step.

With the per time-step output distribution, we compute the probability of different sequences

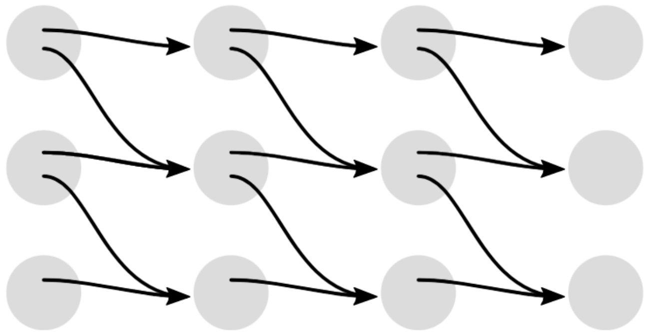
By marginalizing over alignments, we get a distribution over outputs

The algorithm: Loss Function

- The CTC loss can be very expensive to compute.
  - A brute force approach that computes the score for each alignment is expensive
    - There can be a massive number of alignments.
- We can compute the loss faster with a **dynamic programming** algorithm
  - If two alignments have reached the same output at the same step, they can be merged



Summing over all alignments can be very expensive.



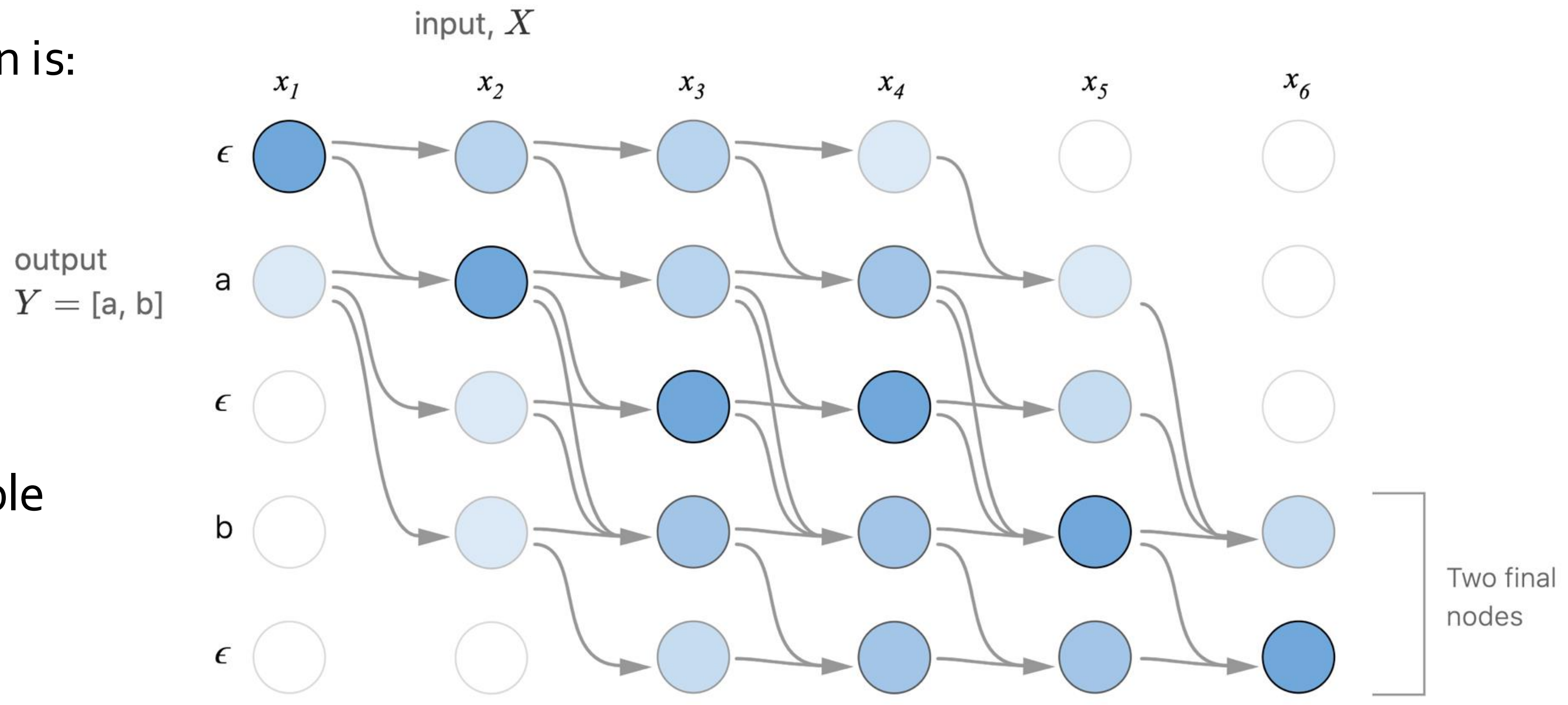
Dynamic programming merges alignments, so it's much faster.

The algorithm: Loss Function

- Example of the computation performed by the dynamic programming algorithm
- Every valid alignment has a path in this graph.
  - For a training set D, the loss function is:

$$\sum_{(X,Y)\in\mathcal{D}} -\log\ p(Y \mid X)$$

  - The CTC loss function is differentiable since it's just sums and products of probabilities



Node (s,t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

The algorithm: Inference

- Find a likely output for a given input by solving:

$$Y^* = \underset{Y}{\operatorname{argmax}}\ p(Y \mid X)$$

- Need to settle for an approximate solution, too expensive to search for the true max
- One heuristic is to take the most likely character at each output

The algorithm: Inference

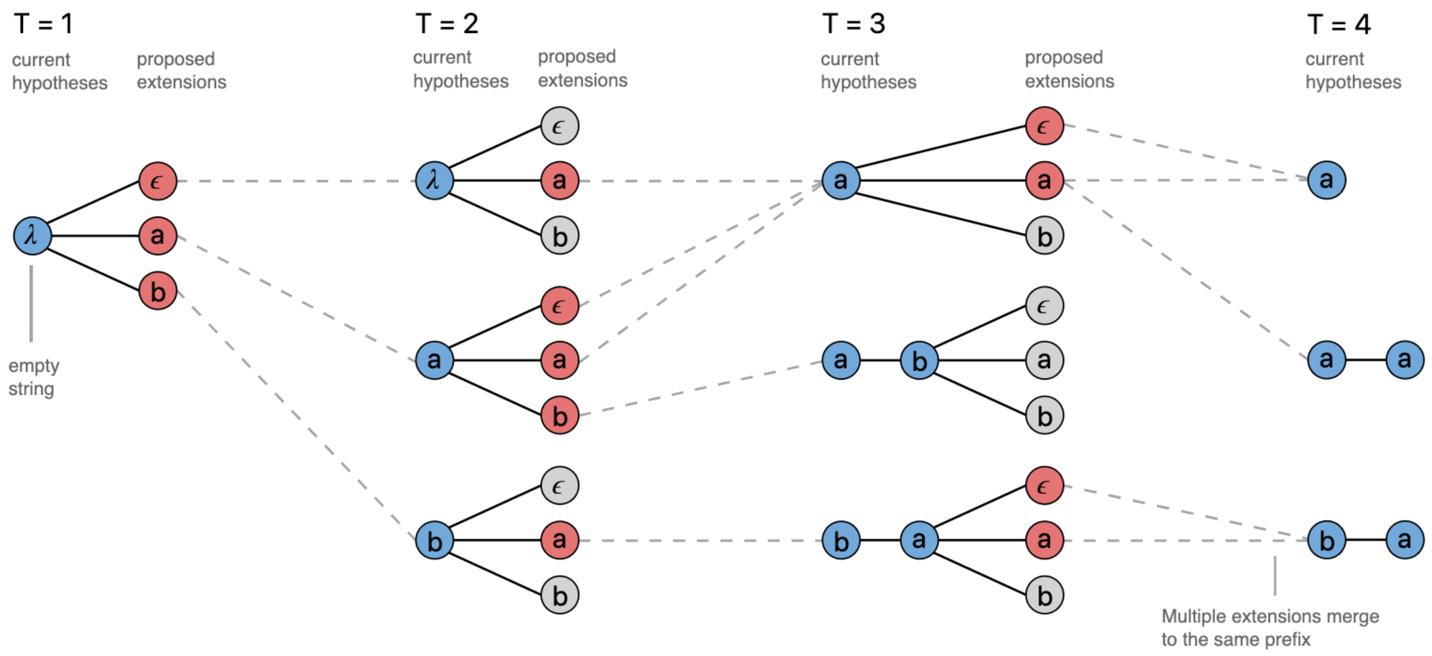
$$Y^* = \underset{Y}{\operatorname{argmax}}\ p(Y \mid X)$$

- Problems?
  - Does not take into account that the same output Y could be produced by two different alignments
  - [a,a] and [a,a,a] individually have lower probability than [b,b], but combined higher and they collapse to [a]
  - With this heuristic, [b] gets picked

The algorithm: Inference

- A better heuristic is to use modified beam search
- Can exchange speed for asymptotically better solution

$$Y^* = \underset{Y}{\operatorname{argmax}}\ p(Y \mid X)$$



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