Yandex

SCHOOL OF DATA ANALYSIS

ASR IV: Context Modelling, Language Models and Attention-based ASR

Andrey Malinin

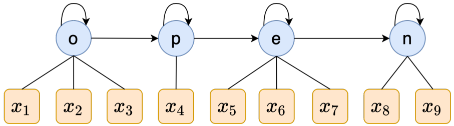
28nd March 2022

Story so far

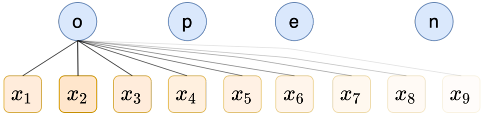
In this previous episode...

Recap - Data Processing and Alignment

- Process the Audio and Text into convenient representations
  - Transform audio into sequence of acoustic features or frames $\boldsymbol{X}_{1:T}$
  - Transform text into a sequence of speech units $\omega_{1:L}$
- Need to dynamically align features $\boldsymbol{X}_{1:T}$ to speech units $\boldsymbol{\omega}_{1:L} \rightarrow$ use:
  - State-Space models (HMMs and CTC)
  - Neural Attention Mechanisms

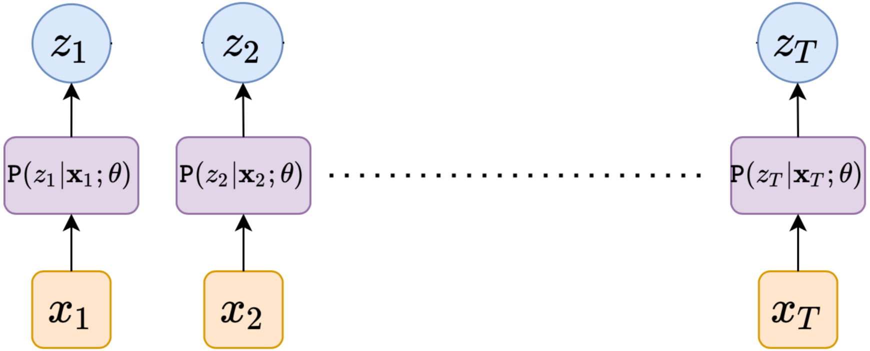


(a) State-Space



(b) Attention Mechanism

Connectionist Temporal Classification (CTC)


z_1
z_2
z_T
$\mathtt{P}(z_1 | \mathbf{x}_1; \theta)$
$\mathtt{P}(z_2 | \mathbf{x}_2; \theta)$
$\mathtt{P}(z_T | \mathbf{x}_T; \theta)$
x_1
x_2
x_T

Connectionist Temporal Classification

- Discriminative State-Space model $\rightarrow$ doesn't model inter-state dependencies

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

- Independently take arg-maxes $\rightarrow$ yields most probable state sequence

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

- GD can still fail to find the best solution $\rightarrow$
  - Grammatical constraints not enforced $\rightarrow$ output 'sounds', but has many errors.
- Use language model to enforce grammatic constraints!

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

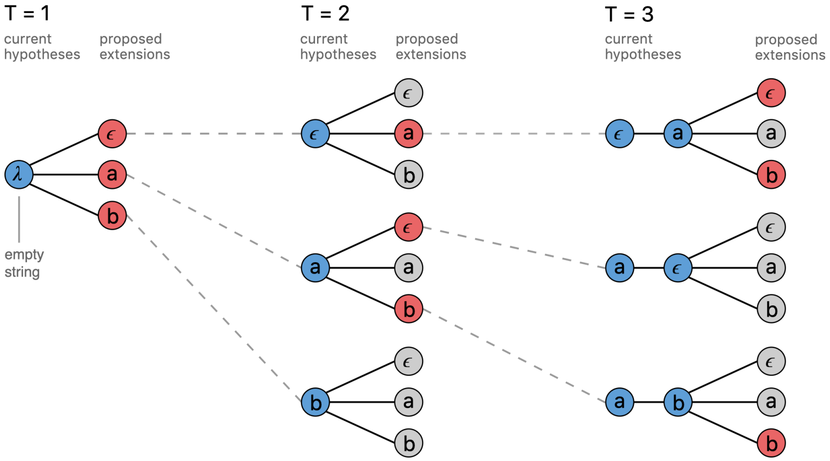
the dynamo electric machine though small was robust for under all the varying speeds of water power and the vicissitudes of the plant to which it belonged it continued in active use until eighteen ninety nine seventeen years

Reference

Example of CTC Output

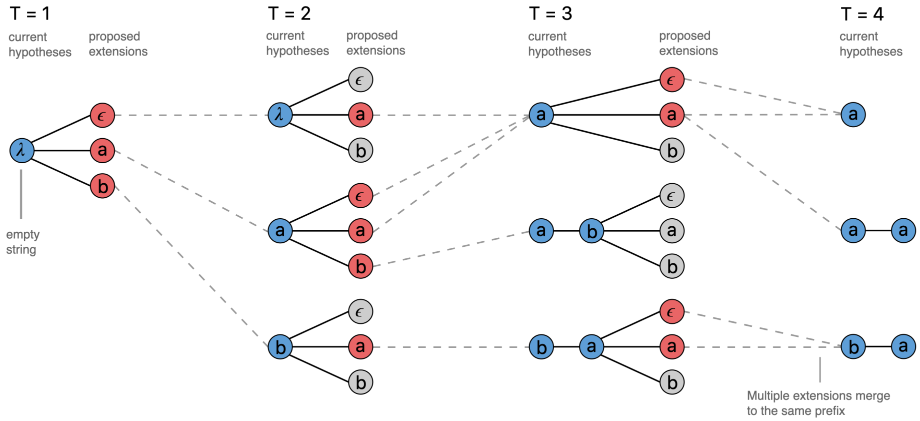
- CTC does not explicitly model output-output interactions
  - Output is ‘phonetically’ close, but orthographically poor
- How can we improve this and enforce grammatical and orthographic constraints?
  - Introduce a **vocabulary** or external word-level **language model**
  - Explicitly model output-output interactions via **RNN-Transducer**

Prefix Beam Search Decoding



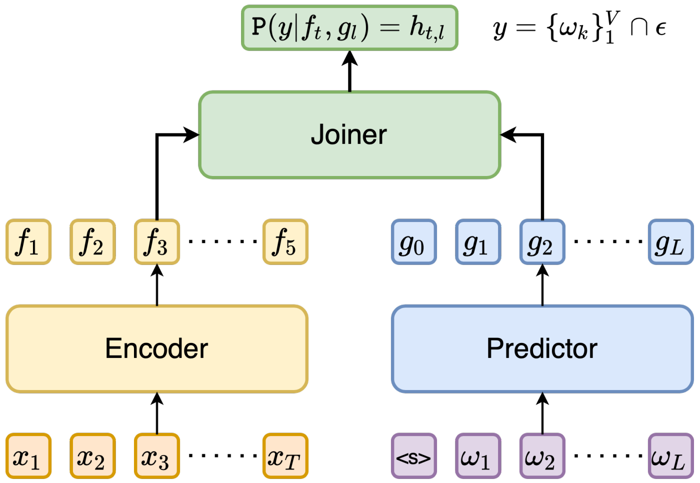
A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.

Prefix Beam Search Decoding



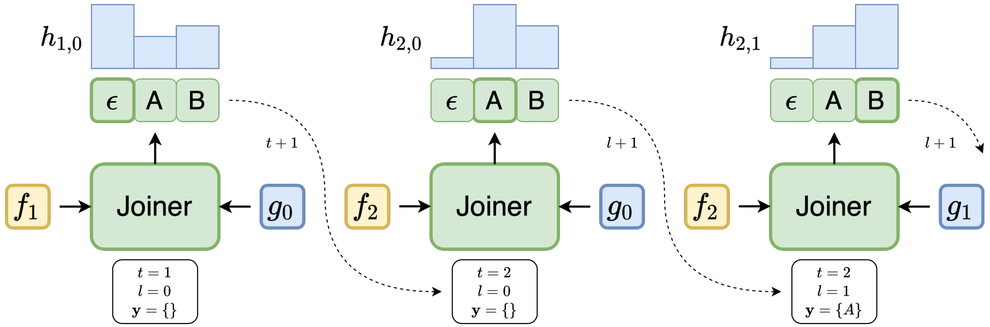
The CTC beam search algorithm with an output alphabet $\{\epsilon, a, b\}$ and a beam size of three.

RNN Transducer - Architecture



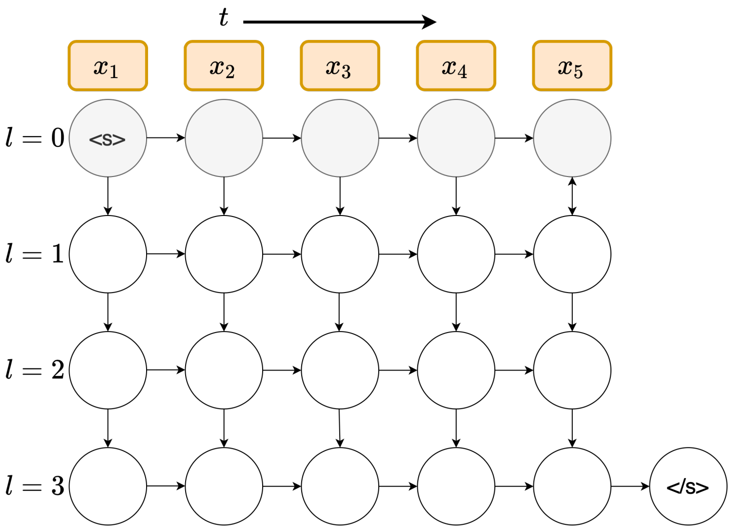
$\mathrm{P}(y|f_t, g_l) = h_{t,l}$
$y = \{\omega_k\}_1^V \cap \epsilon$
Joiner
f_1 f_2 f_3 ...... f_5
g_0 g_1 g_2 ...... g_L
Encoder
Predictor
x_1 x_2 x_3 ...... x_T
<s> ω_1 ω_2 ...... ω_L

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 Alignment Trellis



Language Modelling

How else can we integrate long-range linguistic information?

Language Modelling

Language modelling

Language Modelling

- A language model defines a prior distribution over word sequences:

$$\mathrm{P}(\boldsymbol{w}) = \prod_{q=1}^{Q} \mathrm{P}(w_q | \boldsymbol{w}_{1:q-1})$$

- LMs help us discriminate between different acoustically plausible hypotheses:
  - Ex: "Wreck a nice beach" vs. "Recognize speech"
- LMs can also be defined at the character level or over BPE tokens
  - Choose appropriate context level based on task, language and amount of data
- Two common classes of language models:
  - N-Gram LMs $\rightarrow$ lightweight, cheap, limited flexibility
  - Neural LMs $\rightarrow$ expensive, powerful, expressive

Language Modelling - Evaluating LM Quality

- A language models are typically evaluated in terms of perplexity
  - Important - perplexity is evaluated using Base-2 logarithms!

$$\mathrm{P} = 2^{\mathrm{NLL}}, \ \mathrm{NLL} = \frac{1}{Q}\sum_{q=1}^{Q} -\log_2 \mathrm{P}(w_q|\mathbf{w}_{<q}))$$

- Perplexity is the effective branching factor or confusion of your model
  - Best perplexity of 1, worst perplexity is vocab size $|V|$
- Perplexity of a data $\mathcal{D} = \{\mathbf{w}^{(n)}\}_{n=1}^{N}$ is defined as follows

$$\mathrm{P}(\mathcal{D}) = 2^{\mathrm{NLL}}, \ \mathrm{NLL} = \frac{1}{\sum_{n=1}^{N} Q^{(n)}}\sum_{n=1}^{N}\sum_{q=1}^{Q} -\log_2 \mathrm{P}(w_q^{(n)}|\mathbf{w}_{<q}^{(n)}))$$

- However, it is also important to evaluate LMs on downstream tasks, such as ASR

Language Modelling - N-Gram Language Models

- N-Gram languag models make an N-th order Markov assumption:
  - Unigrams assume words are independent of each other
  - Bigrams condition only on the previous word
  - Trigrams condition on the last two words
- Bigrams are defined as

$$\mathrm{P}_2(\boldsymbol{w}) = \prod_{q=1}^{Q} \mathrm{P}(w_q | \boldsymbol{w}_{1:q-1}) \approx \prod_{q=1}^{Q} \mathrm{P}(w_q | w_{q-1})$$

- General N-Grams are defined as

$$\mathrm{P}_3(\boldsymbol{w}) = \prod_{q=1}^{Q} \mathrm{P}(w_q | \boldsymbol{w}_{1:q-1}) \approx \prod_{q=1}^{Q} \mathrm{P}(w_q | \boldsymbol{w}_{q-N+1:q-1})$$

Language Modelling - N-Gram Language Models

- Given a text corpus, N-Grams probabilities are obtained via via ratio of counts:

$$\mathrm{P}(w_q | w_{q-2}, w_{q-1}) = \frac{\mathrm{Count}(w_{q-2}, w_{q-1}, w_q)}{\mathrm{Count}(w_{q-2}, w_{q-1})}$$

- This has obvious issues - what if certain work combinations never appear?
  - Solution 1: Discouting - re-allocate counts to rare events
  - Solution 2: Back-off - lower the n-gram order and reduce context
- Typically both are combined together

Language Modelling - Discounting and Back-off

- Kneser-Ney smoothing combines discount with back-off. For example for tri-grams

$$\mathrm{P}(w_q | w_{q-2}, w_{q-1}) = \begin{cases} \dfrac{\mathrm{C}(w_{q-2}, w_{q-1}, w_q) - \delta}{\mathrm{C}(w_{q-2}, w_{q-1})} & \text{if } \mathrm{C}(w_{q-2}, w_{q-1}, w_q) > \delta) \\ \alpha(w_q, w_{q-1}) \mathrm{P}(w_q | w_{q-1}) & \end{cases}$$

- Back-off weight α ensures that $\sum_{k=1}^{V} \mathrm{P}(w_q = k | w_{q-2}, w_{q-1}) = 1$
- Back-off can be done recursively with different δ for every level of back-off.

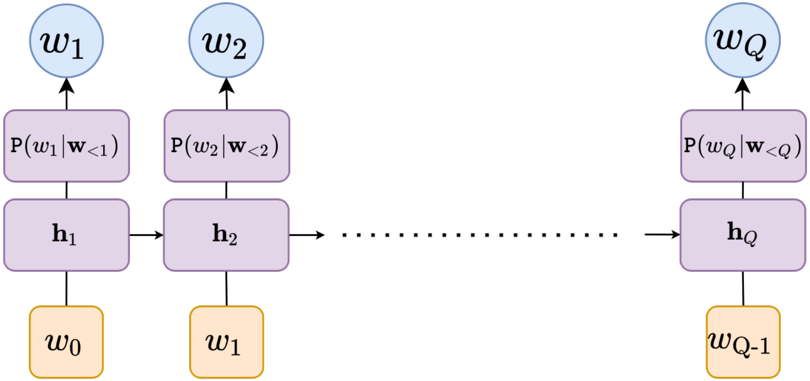
Language Modelling - Software Packages

- Standard implementations of N-Gram language models can easily be found online
  - KenLM
  - SRILM
- For many, python packages exist

Language Modelling - Limitations of N-Grams

- N-Grams have many advantages
  - Fast to train, fast to evaluate via lookup
  - Small N-Grams (Bigram or Trigram) is good for ASR decoder
  - Conceptually simple
- However, N-Grams have several important limitations:
  - Depend only on surface form $\rightarrow$ does not see word similarity
  - Data inefficiency $\rightarrow$ similar words in similar context make different n-grams
  - Limited context $\rightarrow$ longer context requires far more data
- Neural LMs overcome many of these limitation

Language Modelling - Neural Language Models (NLMs)


w_1
w_2
w_Q
$\mathrm{P}(w_1|\mathbf{w}_{<1})$
$\mathrm{P}(w_2|\mathbf{w}_{<2})$
$\mathrm{P}(w_Q|\mathbf{w}_{<Q})$
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_Q$
w_0
w_1
w_{Q-1}

- NLMs express distribution over words as function of previous word and context

Language Modelling - Neural Language Models (NLMs)

- NLMs express distribution over words as function of previous word and context

$$\mathrm{P}(\boldsymbol{w};\boldsymbol{\theta}) = \prod_{q=1}^{Q} \mathrm{P}(w_q|\boldsymbol{w}_{<q};\boldsymbol{\theta}) \approx \prod_{q=1}^{Q} \mathrm{P}(w_q|\boldsymbol{h}_q;\boldsymbol{\theta}) = \prod_{q=1}^{Q} \mathrm{P}(w_q|w_{q-1},\boldsymbol{h}_{q-1};\boldsymbol{\theta})$$

- Can consider recurrent (RNN, GRU, LSTM) or transformer architectures
  - Quality of context modelling depends on architecture, regularization, etc..
- NLMs are typically trained using using cross-entropy loss (max-likelihood)

$$\mathcal{L}(\boldsymbol{\theta},\mathcal{D}) = \frac{1}{\sum_{n=1}^{N} Q^{(n)}} \sum_{n=1}^{N} \sum_{q=1}^{Q^{(n)}} -\ln \mathrm{P}(w_q^{(n)}|\boldsymbol{w}_{<q}^{(n)};\boldsymbol{\theta})$$

- NLMs are more expressive than n-gram, but far more expensive to evaluate

Language Modelling - Language Model Fusion

- Language models are very useful for speech recognition!
  - Beam-search decoding
  - N-best list re-scoring
- **LM Fusion** - LMs can be combined with acoustic models on many levels
  - External LM
  - Integration of level of Neural Network architecture
- Choice of fusion level depends on architecture, language and data

Prefix Decoding

Language Model for improving CTC Prefix-Search decoding

Prefix Decoding - Integration of LM with CTC

- If we consider a generative model (HMM), then LMs naturally integrate:

$$\boldsymbol{w}* = \arg\max_{\boldsymbol{w}} \mathrm{P}(\boldsymbol{X}_{1:T}|\boldsymbol{w}) P(\boldsymbol{w})^{\alpha}$$

- As CTC is a discriminative model, LMs can only be integrated as a heuristic:

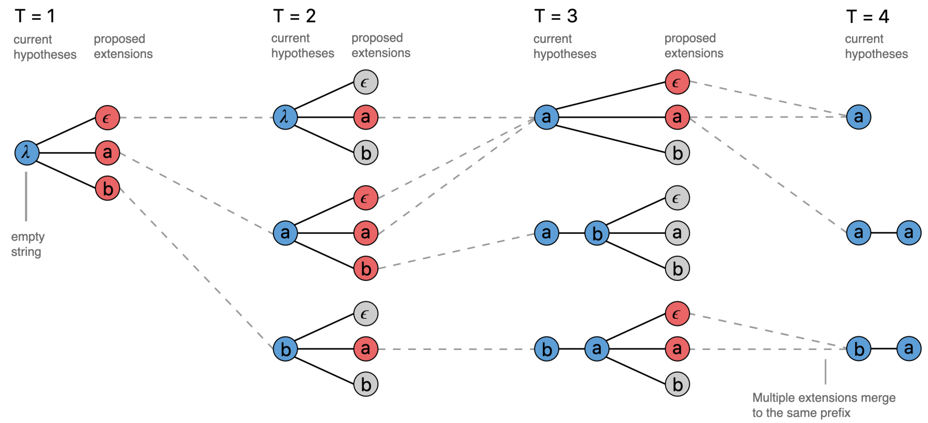
$$\boldsymbol{w}* = \arg\max_{\boldsymbol{w}} \underbrace{\mathrm{P}(\boldsymbol{w}|\boldsymbol{X}_{1:T})}_{\text{CTC prob}} \cdot \underbrace{P(\boldsymbol{w})^{\alpha}}_{\text{LM prob}} \cdot \underbrace{|\boldsymbol{w}|^{\beta}}_{\text{Length Correction}}$$

- LM biases longer sentences to have smaller probability $\rightarrow$ correct with length term
  - Here α de-weights the LM, while β up-weights the length correction term
  - α and β are chosen via cross-validation

Prefix Beam Search Decoding - Recap

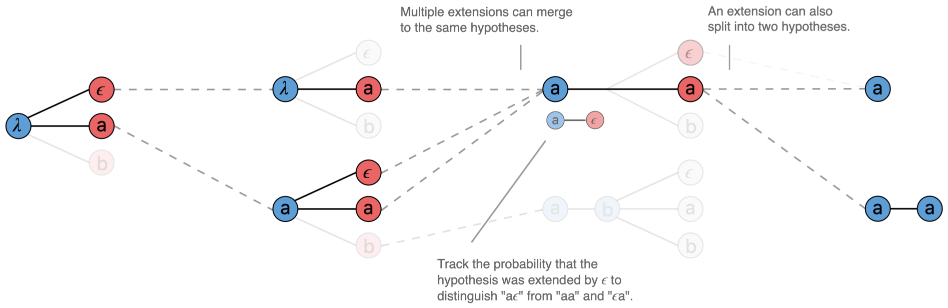
- In CTC, many paths $\pi_{1:T}$ map to the same character sequence $\omega_{1:L}$
  - Greedy Decoding finds only most likely path $\pi^*_{1:T}$
  - Most likely character sequence $\omega^*_{1:L}$ could be composed of many less-likely paths!
- Prefix decoding considers probabilities of multiple paths 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



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

Prefix Beam Search Decoding



Multiple extensions can merge to the same hypotheses.
An extension can also split into two hypotheses.
Track the probability that the hypothesis was extended by ϵ to distinguish "aϵ" from "aa" and "ϵa".
λ
ϵ
a
b

- We are extending the prefix → 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 $\mathtt{Pb}_0(\mathtt{l}) = 1$ and $\mathtt{Pnb}_0(l) = 0$
  - 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:

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

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:

$$\begin{aligned} \mathtt{Pnb}_t(l+c) &= \mathtt{P}(C|\boldsymbol{x}_t) \cdot \mathtt{Pb}_{t-1}(l) \\ \mathtt{Pnb}_t(l) &= \mathtt{P}(C|\boldsymbol{x}_t) \cdot \mathtt{Pnb}_{t-1}(l) \end{aligned}$$

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:

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

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

- We want to apply LM only when we have a new complete word
  - Apply LM if chatacter C is a non-repeat space
- If c is space

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

- If is any other non-repeat, non-space non-blank character

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

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

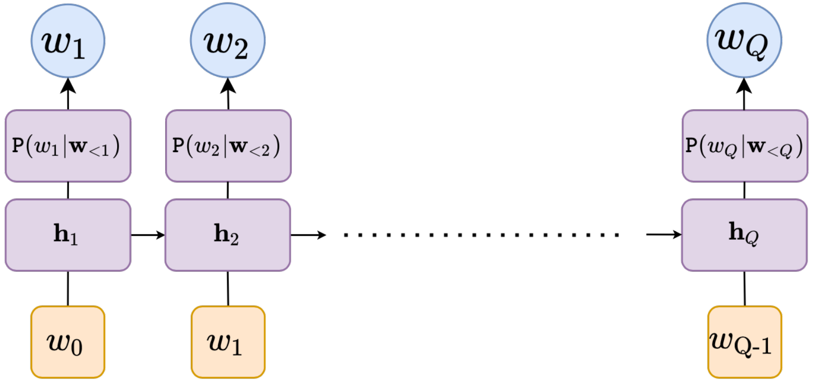
5 Mins Break!

Break!

Autoregressive Attention-based Models

Autoregressive Attention-based Models

Language Modelling - Neural Language Models (NLMs)



- NLMs express distribution over words as function of previous word and context

Language Modelling - Neural Language Models (NLMs)

- NLMs express distribution over words as function of previous word and context

$$\mathrm{P}(\boldsymbol{w};\boldsymbol{\theta}) = \prod_{q=1}^{Q} \mathrm{P}(w_q|\boldsymbol{w}_{<q};\boldsymbol{\theta}) \approx \prod_{q=1}^{Q} \mathrm{P}(w_q|\boldsymbol{h}_q;\boldsymbol{\theta}) = \prod_{q=1}^{Q} \mathrm{P}(w_q|w_{q-1},\boldsymbol{h}_{q-1};\boldsymbol{\theta})$$

- Neural LMs are typically trained using using cross-entropy loss (max-likelihood)

$$\mathcal{L}(\boldsymbol{\theta},\mathcal{D}) = \frac{1}{\sum_{n=1}^{N} Q^{(n)}} \sum_{n=1}^{N} \sum_{q=1}^{Q^{(n)}} -\ln \mathrm{P}(w_q^{(n)}|\boldsymbol{w}_{<q}^{(n)};\boldsymbol{\theta})$$

- Can consider recurrent (RNN, GRU, LSTM) or transformer architectures

Autoregressive Attention-based Enoder-Decoder Models

NLMs can generate language... Can we condition them on audio?

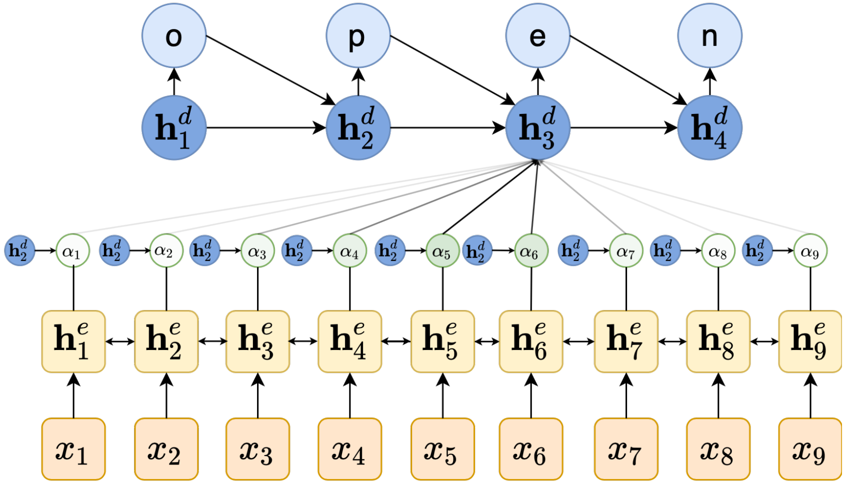
Autoregressive Attention-based Enoder-Decoder Models

- Autoregressive attention-based ASR $\rightarrow$ ASR via conditional LM

$$\mathrm{P}(\boldsymbol{w}_{1:Q}|\boldsymbol{X}_{1:T}) = \prod_{l=1}^{L} \mathrm{P}(w_q|\boldsymbol{w}_{<q}, \boldsymbol{X}_{1:T})$$

- Attention-based ASR systems have three main components
  - Module for generating text - the conditional NLM or Decoder
  - Module for processing and compressing audio - the Encoder
  - Module for aligning text and audio - Attention Mechanism
- Directly integrates LM and conditions on the acoustics
  - $+$ Jointly trains all components of ASR system
  - $+$ Mathematically simpler formulation (training, beam-search, alignment)
  - - Needs MUCH more data.

Autoregressive Attention-based Models



Decoder

- The decoder is an NLM which generates text conditioned on the audio and history

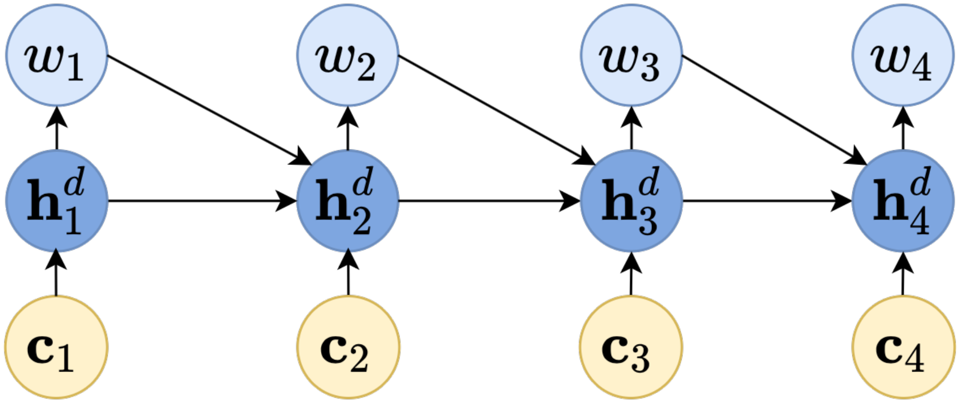
$$\mathrm{P}(\boldsymbol{w}_{1:Q}|\boldsymbol{X}_{1:T};\boldsymbol{\theta}) = \prod_{l=1}^{L} \mathrm{P}(w_q|\boldsymbol{w}_{<q},\boldsymbol{X}_{1:T};\boldsymbol{\theta}) = \prod_{l=1}^{L} \mathrm{P}(w_q|w_{q-1},\boldsymbol{h}_{q-1},\boldsymbol{c}_q;\boldsymbol{\theta}))$$

- Decoder is conditioned on previous word w_{q-1}, history $\boldsymbol{h}_{q-1}$ and audio-context $\boldsymbol{c}_q$
  - History vector $\boldsymbol{h}_{q-1}$ encodes the previously generated context
  - Audio-context $\boldsymbol{c}_q$ is a representation of $\boldsymbol{X}_{1:T}$ appropriate for generating next word
  - Audio-context $\boldsymbol{c}_q$ is provided by the attention mechanism and encoder
- Can generate a sentence either via sampling or beam-search

$$\boldsymbol{w}^*_{1:Q} = \arg\max_{\boldsymbol{w}_{1:Q}} \mathrm{P}(\boldsymbol{w}_{1:Q}|\boldsymbol{X}_{1:T};\boldsymbol{\theta})$$

- Training and Evaluation are mismatched!
  - Training - reference context. Evaluation - generated context!

Decoder - Recurrent Decoder



- Recurrent Decoders typically implemented as RNN, GRU or LSTM
  - Can alternatively consider Transformer-based architectures

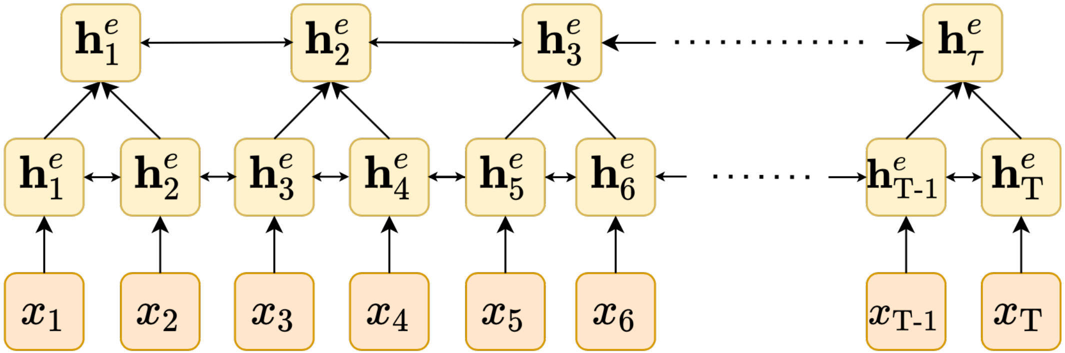
Encoder

- Input mel-spectrogram features $\boldsymbol{X}_{1:T}$ are processed via an encoder

$$\boldsymbol{H}^e_{1:\tau} = \{\boldsymbol{h}^e_1, \cdots, \boldsymbol{h}^e_\tau\} = \mathbf{f}(\boldsymbol{X}_{1:T}; \boldsymbol{\theta})$$

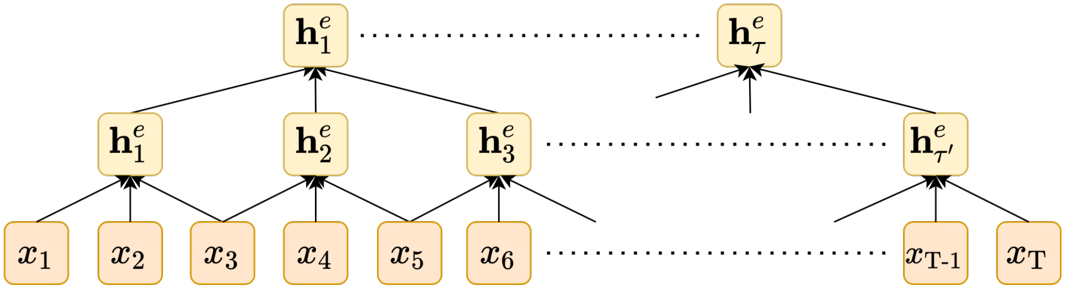
- The encoder has two goals
  - Provide a convenient presentation for further processing
  - Improve representation robustness
  - Reduce the time-resolution of the input features - $\tau << T$
- Commonly used architectures are
  - Pyramidal BiLSTM
  - Dilated Convolutions
  - Transformer-based self-attention architecutres
  - Combination of the above (DL is Lego!)

Encoder - Pyramidal BiLSTM



- Pyramidal BiLSTMs are an expressive, but expensive and slow encoder
  - Have to process sequentially, many parameters

Encoder - Dilated Convolutions



- Dilated Convolutions are lighted, faster and more parallelizable
  - Can add residual connections and layer norm before next layer

Attention Mechanism

- Attention ‘focuses’ on part of input encoding relevant for predicting next word.
- Express context $\boldsymbol{c}_q$ as a convex combination of encoder outputs $\boldsymbol{h}^e_{1:T}$

$$\boldsymbol{c}_q = \sum_{t=1}^{T} \alpha_t^q \boldsymbol{h}_t^e, \quad \sum_{t=1}^{T} \alpha_t^q = 1, \alpha_t^q \geq 0$$

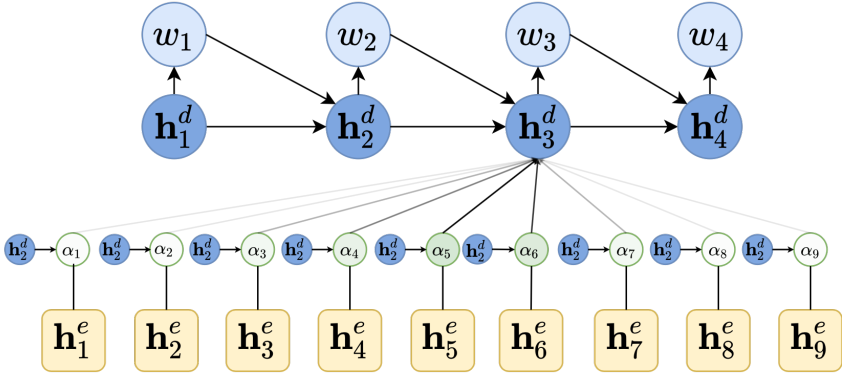
- Attention weights α_t^q are a function of the history $\boldsymbol{w}_{1:q-1}$ and encoder features $\boldsymbol{h}^e_t$.

$$\alpha_t^q = \frac{e^{z_t^q}}{\sum_{t=1}^{T} e^{z_t^q}}, \quad z_t^q = g(\boldsymbol{h}^d_{q-1}, \boldsymbol{x}_t).$$

- The function $g(\boldsymbol{h}^d_{l-1}, \boldsymbol{h}^e_t)$ is typically a small neural network with 1 layer

$$g(\boldsymbol{h}^d_{q-1}, \boldsymbol{h}^e_t) = \boldsymbol{z}^{\mathrm{T}} \tanh(\boldsymbol{W}^d \boldsymbol{h}^d_{q-1} + \boldsymbol{W}^e \boldsymbol{h}^e_t + \boldsymbol{b})$$

Autoregressive Attention-based Models



Training and Inference

- Encoder-Decoder models are trained using using teacher-forcing max-likelihood
  - Maximize likelihood of target word $w_q^{(n)}$ given reference context $\boldsymbol{w}_{<q}^{(n)}$!
  - Do not need to know alignment - handled by attention-mechanism

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \frac{1}{\sum_{n=1}^{N} Q^{(n)}} \sum_{n=1}^{N} \sum_{q=1}^{Q^{(n)}} - \ln \mathrm{P}(w_q^{(n)} | \boldsymbol{w}_{<q}^{(n)}, \boldsymbol{X}_{1:T^{(n)}}^{(n)}; \boldsymbol{\theta})$$

- Can generate a sentence either via sampling or beam-search

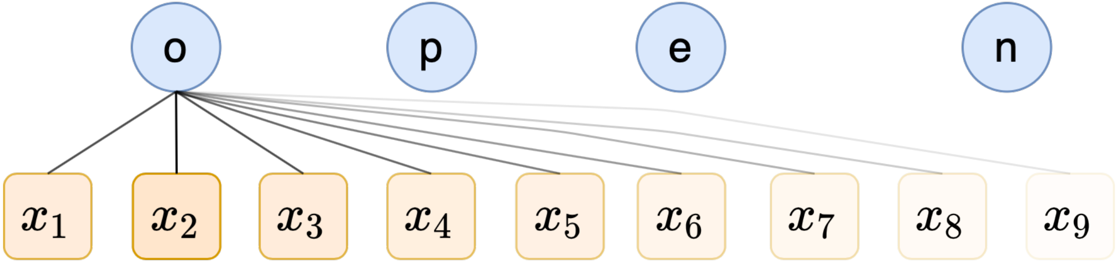
$$\boldsymbol{w}_{1:Q}^{*} = \arg\max_{\boldsymbol{w}_{1:Q}} \mathrm{P}(\boldsymbol{w}_{1:Q} | \boldsymbol{X}_{1:T}; \boldsymbol{\theta})$$

- Training and Evaluation are mismatched!
  - Do not have reference context during beam-search or sampling!

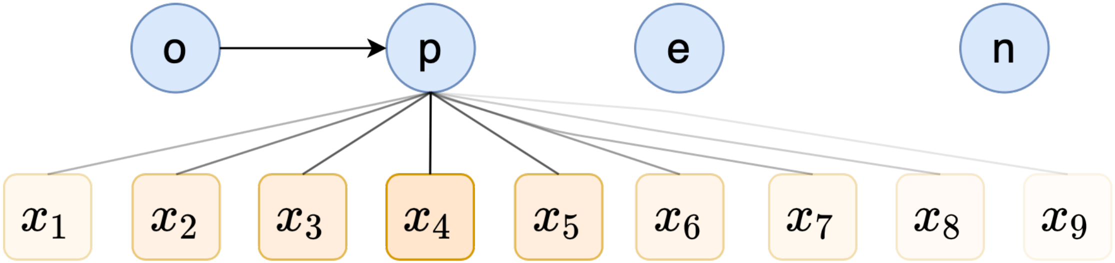
Autoregressive Attention-based Enoder-Decoder Models

Putting it all together

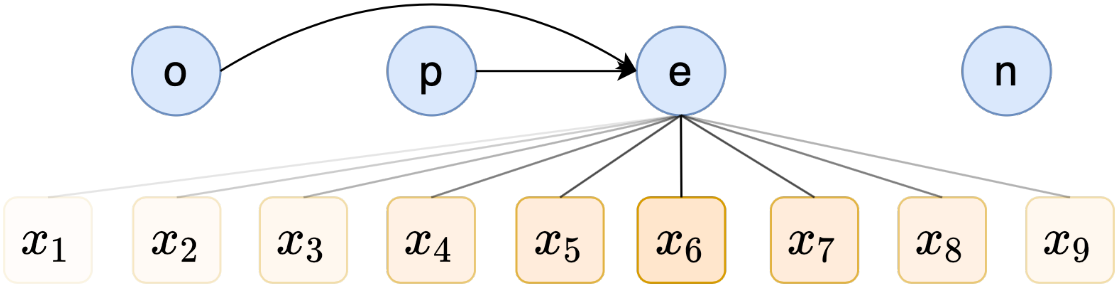
Alignment via Attention Mechanism



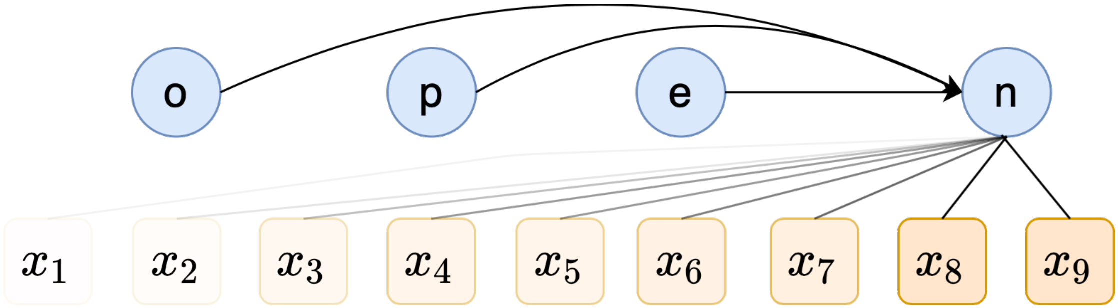
Alignment via Attention Mechanism



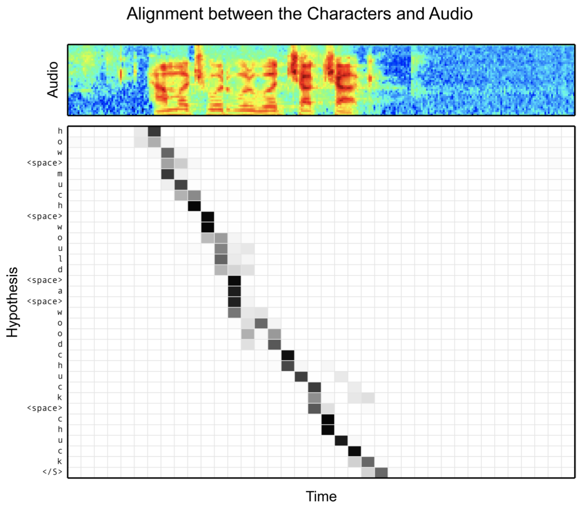
Alignment via Attention Mechanism



Alignment via Attention Mechanism



Attention-based Autoregressive Encoder-Decoder Models



Challenges of Attention-based Encoder-Decoder Models

- Attention-based Encoder-Decoder models face a range of challenges
  - Mismatched training and inference context
  - Attention is non-monotonic and can be diluted if time-resolution too high
  - Cannot use separate, non-paired (with audio) training data for decoder
- Scheduled sampling can help with training/inference mismatch
- Lower time resolution in encoder for attention, or use location-sensitive attention
- Training separate language model and interpolate

Regularization via Scheduled Sampling

- With probability 0.1 use sampled previous word instead of reference in training

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \frac{1}{\sum_{n=1}^{N} Q^{(n)}} \sum_{q=1}^{Q^{(n)}} - \ln \mathrm{P}(w_q^{(n)} | \boldsymbol{\omega}_{q-1}^{(n)}, \boldsymbol{w}_{<q-1}^{(n)}, \boldsymbol{X}_{1:T^{(n)}}^{(n)}; \boldsymbol{\theta})$$

$$\boldsymbol{\omega}_{q-1}^{(n)} = \begin{cases} \boldsymbol{w}_{q-1}^{(n)} & \text{with probability } 0.9 \\ \boldsymbol{w}_{q-1} \sim \mathrm{P}(w_{q-1} | \boldsymbol{w}_{<q-2}^{(n)}, \boldsymbol{X}_{1:T^{(n)}}^{(n)}; \boldsymbol{\theta}) & \text{with probability } 0.1 \end{cases}$$

- Simulates partially incorrect context and makes model more robust to it.

Language Model Interpolation

- Can use a separately trained language model interpolated with ASR system

$$\hat{\mathrm{P}}(w_q^{(n)}|\boldsymbol{w}_{<q}^{(n)}, \boldsymbol{X}_{1:T^{(n)}}^{(n)}; \boldsymbol{\theta}) =$$
$$= \lambda_1 \cdot \mathrm{P}(w_q^{(n)}|\boldsymbol{w}_{<q}^{(n)}, \boldsymbol{X}_{1:T^{(n)}}^{(n)}; \boldsymbol{\theta}) + \lambda_2 \cdot \mathrm{P}(w_q^{(n)}|\boldsymbol{w}_{<q}^{(n)}; \phi), \; \lambda_1 + \lambda_2 = 1$$

- Simple, but often effective approach.

Conclusion

So what have we learned?

Conclusion

- Language Models define a distributions over words/letters/tokens
  - Evaluated using Perplexity, but better using down-stream tasks
  - N-Gram vs Neural LMs
  - LMs help in disambiguation of acoustically similar hypotheses in ASR
- Prefix Decoding with LMs for CTC
- Autoregressive ASR systems consist of three components
  - Decoder - conditional language model
  - Encoder - Pyramidal BiLSTM or Dilated Convolutional network
  - Attention Mechanism - form alignment.
- Mismatched training and inference setup
  - Trained using teacher-forcing max likelihood
  - Inference using beam-search or sampling