End-to-end systems 2: Sequence-to-sequence models

Steve Renals

Automatic Speech Recognition – ASR Lecture 16
14 March 2019

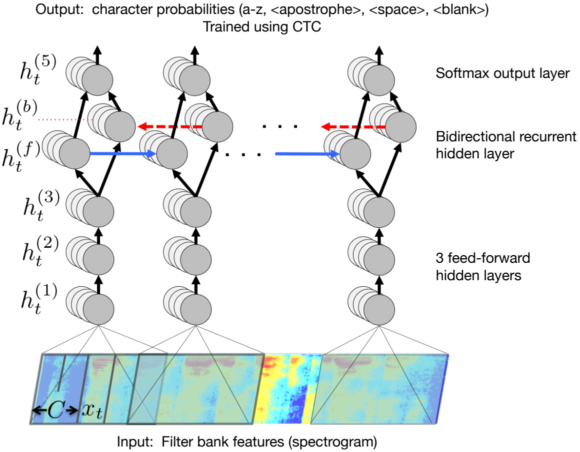
End-to-end systems

- End-to-end systems are systems which learn to directly map from an input sequence X to an output sequence Y, estimating $P(Y|X)$
- ML trained HMMs are kind of end-to-end system – the HMM estimates $P(X|Y)$ but when combined with a language model gives an estimate of $P(Y|X)$
- Sequence discriminative training of HMMs (using GMMs or DNNs) can be regarded as end-to-end
  - But training is quite complicated – need to estimate the denominator (total likelihood) using lattices, first train conventionally (ML for GMMs, CE for NNs) then finetune using sequence discriminative training
  - Lattice-free MMI is one way to address these issues
- Other approaches based on recurrent networks which directly map input to output sequences
  - CTC – Connectionist Temporal Classification
  - Encoder-decoder approaches

End-to-end systems

- End-to-end systems are systems which learn to directly map from an input sequence X to an output sequence Y, estimating $P(Y|X)$
- ML trained HMMs are kind of end-to-end system – the HMM estimates $P(X|Y)$ but when combined with a language model gives an estimate of $P(Y|X)$
- Sequence discriminative training of HMMs (using GMMs or DNNs) can be regarded as end-to-end
  - But training is quite complicated – need to estimate the denominator (total likelihood) using lattices, first train conventionally (ML for GMMs, CE for NNs) then finetune using sequence discriminative training
  - Lattice-free MMI is one way to address these issues
- Other approaches based on recurrent networks which directly map input to output sequences
  - CTC – Connectionist Temporal Classification (last lecture)
  - **Encoder-decoder approaches**

Recap – CTC Deep Speech Architecture



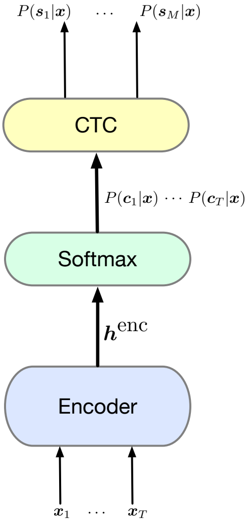
Recap – CTC

- Adds a blank (ϵ) symbol to the output labels
- A deep LSTM (for example) maps input sequence $\boldsymbol{X}$ (length T) to a label sequence $\boldsymbol{C}$ (length T)
- Use CTC compression rule (merge adjacent repeated symbols, then remove blanks) to produce subword sequence $\boldsymbol{S}$ (length $M \leq T$)
- CTC loss function computes the probability $P(\boldsymbol{S})|\boldsymbol{X}$ by summing over all possible valid alignments $P(\boldsymbol{C}|\boldsymbol{X})$

CTC Model

View CTC as having three components:

- **Encoder**: Deep (bidirectional) LSTM recurrent network which maps acoustic features $\boldsymbol{X} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ to a sequence of hidden vectors $\boldsymbol{h}^{\text{enc}} = \boldsymbol{h}_1^{\text{enc}}, \ldots, \boldsymbol{h}_T^{\text{enc}}$.
- **Softmax**: Computes the label probabilities $P(\boldsymbol{c}_1|\boldsymbol{X}), \ldots, P(\boldsymbol{c}_T|\boldsymbol{X})$
- **CTC**: Computes the subword sequence $P(\boldsymbol{s}_1|\boldsymbol{X}), \ldots, P(\boldsymbol{s}_M|\boldsymbol{X})$



Limitations of CTC

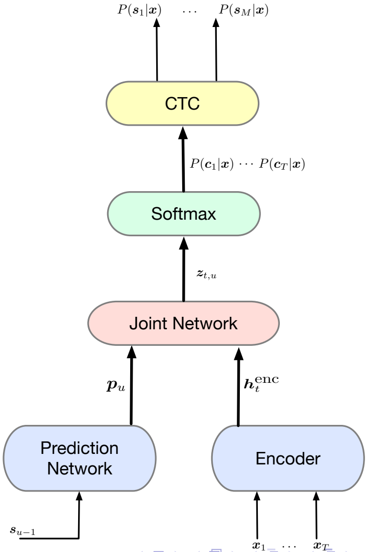
- CTC – pros
  - Can train end-to-end without requiring framewise alignments
  - Sums over all possible alignments (using forward-backward)
  - Preserves monotonic relationship between acoustic frames and output labels

Limitations of CTC

- CTC – pros
  - Can train end-to-end without requiring framewise alignments
  - Sums over all possible alignments (using forward-backward)
  - Preserves monotonic relationship between acoustic frames and output labels
- CTC – cons
  - Assumes output predictions at different times are independent
  - Requires additional language and pronunciation models to introduce dependencies between output label (although it is possible to include a subword-language model in the CTC compression component) (Google experiments)
  - Incorporation of language models is typically ad-hoc
  - End-to-end training of CTC models (also of LF-MMI models) updates the acoustic model parameters using a sequence level criterion, but does not update the pronunciations or language models

RNN Transducer Model

- **Encoder:** Acoustic model network mapping acoustic features $\boldsymbol{X} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ to hidden vectors $\boldsymbol{h}^{\text{enc}} = \boldsymbol{h}_1^{\text{enc}}, \ldots, \boldsymbol{h}_T^{\text{enc}}$.
- **Prediction network**: Recurrent network which takes the previous output subword label $\boldsymbol{s}_{u-1}$ as input and predicts the next subword label $\boldsymbol{p}_u$ – acts as a language model (over subwords)
- **Joint network**: Computes a joint hidden vector $\boldsymbol{Z} = \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T$ by a applying a shallow feed-forward net to $\boldsymbol{h}^{\text{enc}}$ and $\boldsymbol{p}_u$
- Followed by **softmax** and **CTC** components as before



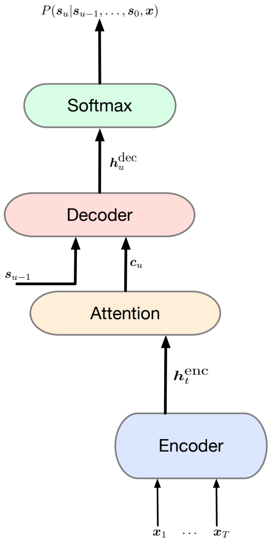
RNN Transducer Model

- RNN transducer can operate left-to-right is a frame-synchronous manner (if the encoder is a unidirectional LSTM)
- Acoustic model (encoder) and language model (prediction network) parts are modelled independently and combined in the joint network. However everything is optimised to a common sequence-level objective (using the CTC loss function).
- With sufficient training data, additional language and pronunciation models are not necessary (Google experiments)
- The recently announced Google "all-neural" on-device speech recognition uses unidirectional RNN transducers

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

Attention-based Encoder-Decoder Model

- **Encoder:** Acoustic model using a recurrent network to map acoustic features $\boldsymbol{X} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ to hidden vectors $\boldsymbol{h}^{\text{enc}} = \boldsymbol{h}^{\text{enc}}_1, \ldots, \boldsymbol{h}^{\text{enc}}_T$.
- **Decoder**: Computes distribution over labels conditioned on previously predicted labels and the acoustics, $P(\boldsymbol{s}_u | \boldsymbol{s}_{u-1}, \ldots, \boldsymbol{s}_0, \boldsymbol{x})$
- **Attention**: Constructs a *context vector* for the decoder network based on attention weights computed over all frames in the encoder output
- Google's "Listen, Attend, and Spell" model: Chan et al (2016), ICASSP. https://ieeexplore.ieee.org/abstract/document/7472621



The Decoder

- The decoder directly generates the output subword sequence $\boldsymbol{S}$
- At each decoding time step u, the decoder RNN uses the previous output $\boldsymbol{s}_{u-1}$, the previous decoder RNN hidden state $\boldsymbol{h}_{u-1}^{\text{dec}}$, and the previous context vector $\boldsymbol{c}_{u-1}$ to generate the current decoder hidden state $\boldsymbol{h}_u^{\text{dec}}$

$$\boldsymbol{h}_u^{\text{dec}} = \text{RNN}(\boldsymbol{h}_{u-1}^{\text{dec}}, \boldsymbol{s}_{u-1}, \boldsymbol{c}_{u-1})$$

- The context vector is computed by the attention mechanism

The Attention Mechanism

- The attention mechanism uses the current decoder RNN hidden state $\boldsymbol{h}^{\text{dec}}_{u-1}$, and the sequence of encoder hidden states $\boldsymbol{h}^{\text{enc}}$ to compute an alignment vector $\boldsymbol{\alpha}_i$:

$$\alpha_{ut} = \text{Attention}(\boldsymbol{h}^{\text{dec}}_u, \boldsymbol{h}^{\text{enc}}_t)$$

- The alignment vector is used as weights in a weighted sum of the encoder hidden states to compute the context vector $\boldsymbol{c}_u$:

$$\boldsymbol{c}_u = \sum_{t=1}^{T} \alpha_{ut} \boldsymbol{h}^{\text{enc}}_t$$

- The decoder uses the context vector $\boldsymbol{c}_u$ and the current decoder hidden state $\boldsymbol{h}^{\text{dec}}_u$ to estimate the subword distribution:

$$\boldsymbol{s}_u \sim \text{LabelDistribution}(\boldsymbol{c}_u, \boldsymbol{h}^{\text{dec}}_u)$$

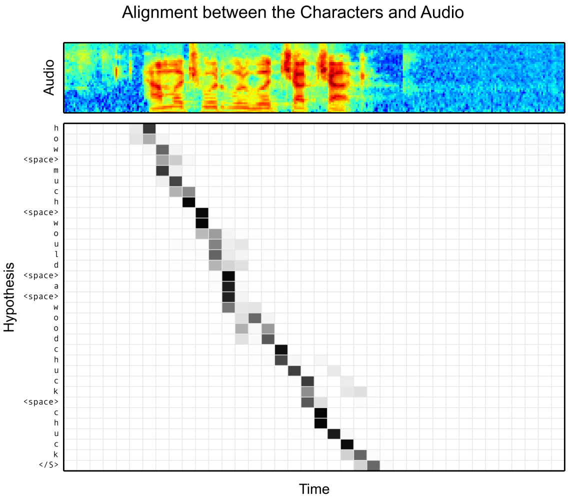
  where LabelDistribution is a single layer neural network with a softmax output over the labels.

Alignment Vector

- Attention models the alignment between the current output $\boldsymbol{s}_u$ and the input sequence $\boldsymbol{x}$ – it matches the "input clock" with the "output clock"
- Various ways to compute the attention - content-based attention commonly used. Single hidden layer followed by a softmax

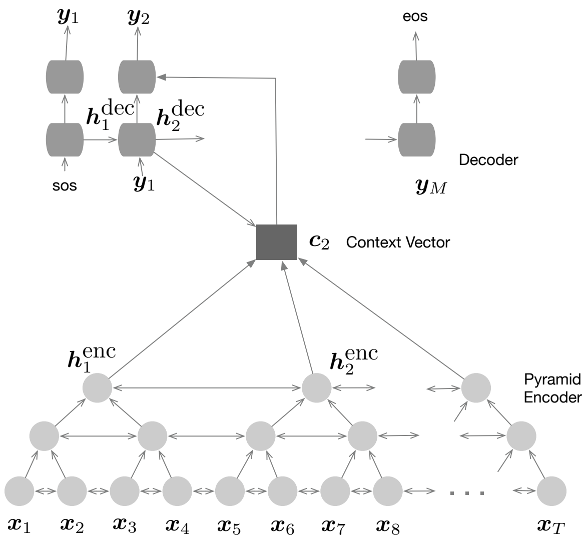
$$e_{ut} = \boldsymbol{v}^T \tanh(\boldsymbol{W}\boldsymbol{h}_u^{\text{dec}} + \boldsymbol{V}\boldsymbol{h}_t^{\text{enc}} + \boldsymbol{b})$$
$$\alpha_{ut} = \frac{\exp(e_{ut})}{\sum_k \exp(e_{uk})}$$

Alignment between labels and acoustics

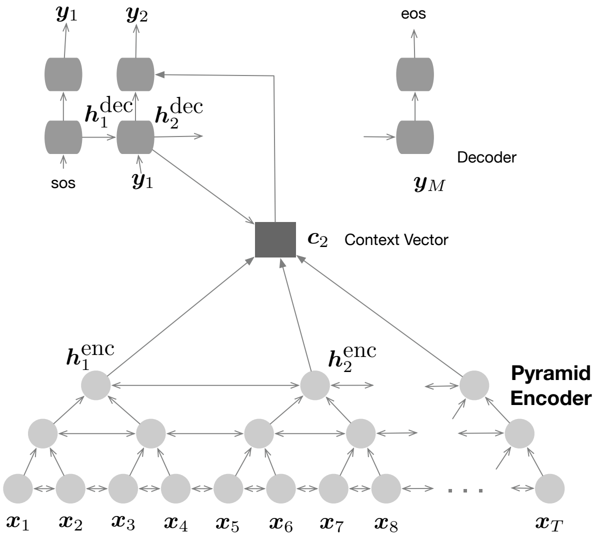


“How much would a woodchuck chuck”

Attention Mechanism



Attention Mechanism



Pyramid Encoder

- A significant problem with a naive end-to-end model is the length of the input sequences... A direct BLSTM encoder can be difficult and slow to train – hard to extract the relevant information from many time steps
- Use a pyramid architecture – each successive layer reduces the resolution by a factor of 2.
  - Typical deep BLSTM hidden state (layer j, time t):

$$\boldsymbol{h}_t^j = RNN(h_t^{j-1}, h_{t-1}^j)$$

  - Pyramid model concatenates consecutive hidden states:

$$\boldsymbol{h}_t^j = pyrRNN([h_{2t}^{j-1}, h_{2t+1}^{j-1}], h_{t-1}^j)$$

  - 3 layers in a pyramid architecture reduces the time resolution (shortens the sequence) by a factor of 8
  - The attention mechanism thus has an easier job, weighting over 8x fewer encoder hidden states

Learning

- Model trained to maximise the log probability of correct sequences

$$\sum_u \log P(\boldsymbol{s}_u | \boldsymbol{x}, \boldsymbol{s}_{<u})$$

  where $\boldsymbol{s}_{<u}$ is the sequence $s_1, \ldots, s_{u-1}$
- An interesting subtlety: what value should be used for $\boldsymbol{s}_{<u}$?
  - The previous predictions? this is consistent between training and test, but adds noise at training time

Learning

- Model trained to maximise the log probability of correct sequences

$$\sum_u \log P(\boldsymbol{s}_u | \boldsymbol{x}, \boldsymbol{s}_{<u})$$

  where $\boldsymbol{s}_{<u}$ is the sequence $s_1, \ldots, s_{u-1}$
- An interesting subtlety: what value should be used for $\boldsymbol{s}_{<u}$?
  - The previous predictions? this is consistent between training and test, but adds noise at training time
  - The ground truth labels (*teacher forcing*)? This speeds up learning, especially early on, but there is a mismatch between training and testing

Learning

- Model trained to maximise the log probability of correct sequences

$$\sum_{u} \log P(\boldsymbol{s}_u | \boldsymbol{x}, \boldsymbol{s}_{<u})$$

  where $\boldsymbol{s}_{<u}$ is the sequence $s_1, \ldots, s_{u-1}$
- An interesting subtlety: what value should be used for $\boldsymbol{s}_{<u}$?
  - The previous predictions? this is consistent between training and test, but adds noise at training time
  - The ground truth labels (*teacher forcing*)? This speeds up learning, especially early on, but there is a mismatch between training and testing
  - **Scheduled sampling**? Sample a label from the estimated distribution. This reduces the noise in training, but doesn't create a big gap between training and test

Decoding and Rescoring

- Decode without a separate pronunciation model or an external language model!
- Simply decode the grapheme sequence! (It is possible to rescore with a language model if desired)
- Decoding use a beam search in which 15-best hypotheses are retained at each decoding step

Results (2017)

Google Voice Search data, 12,500h training data, 15M hand-transcribed utterances

Model	Clean		Noisy		numeric
	dict	vs	dict	vs	
Baseline Uni. CDP	6.4	9.9	8.7	14.6	11.4
Baseline BiDi. CDP	5.4	8.6	6.9	-	11.4
End-to-end systems					
CTC-grapheme[3]	39.4	53.4	-	-	-
RNN Transducer	6.6	12.8	8.5	22.0	9.9
RNN Trans. with att.	6.5	12.5	8.4	21.5	9.7
Att. 1-layer dec.	6.6	11.7	8.7	20.6	9.0
Att. 2-layer dec.	**6.3**	**11.2**	**8.1**	**19.7**	**8.7**

Prabhavalkar et al (2017), "A Comparison of Sequence-to-Sequence Models for Speech Recognition", Interspeech. https://www.isca-speech.org/archive/Interspeech_2017/abstracts/0233.html

Other Refinements (2018)

- Wordpiece models – rather than using single graphemes as labels use multi-grapheme units (up to a word in length) - similar to bye pair encoding in machine translation
- Multiheaded attention – use multiple attention distributions
- Minimum WER training – modify the loss function to interpolate a word error rate term
- Label smoothing – smooth the ground truth distribution by interpolating with a uniform distribution
- LM rescoring – use an external language model (5-gram) trained on large amount of text

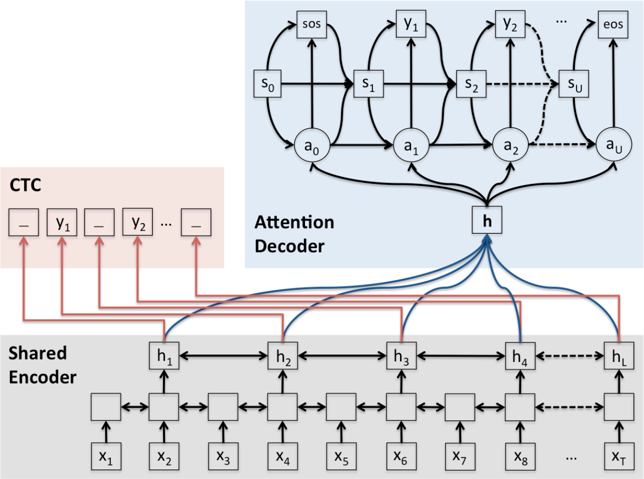
Reduced WER on Voice Search from 9.2% to 5.6% – their hybrid HMM-LSTM system has WER of 6.7% on this task

Chiu et al, "State-of-the-art sequence recognition with sequence-to-sequence models", ICASSP 2018.
https://arxiv.org/abs/1712.01769

Hybrid CTC/Attention

- Attention is very flexible – does not constrain relationship between acoustics and labels to be monotonic
- This can be a problem, especially when huge amounts of training data not available
- Possible solutions:
  - Windowed attention, in which the attention is restricted a set of encoder hidden states
  - Hybrid CTC/Attention model - use CTC and attention jointly during training and recognition – regularises the system to favour monotonic alignments

Hybrid CTC/Attention



Watanabe et al (2017), "Hybrid CTC/Attention Architecture for End-to-End Speech Recognition", IEEE STSP, 11:1240–1252. https://ieeexplore.ieee.org/document/8068205

Summary

- End-to-end models for speech recognition: CTC, RNN Transducer, Attention Encoder-Decoder
- RNN Transducer and Attention-based model integrate acoustic model, pronunciation model, and language model into a single neural network
- With large amounts of hand-transcribed training data, attention-based model can be more accurate than context-dependent NN/HMM
- RNN transducer can operate in online (left-to-right) mode
- Attention based model operates over an utterance at a time (since attention is over the complete encoded utterance)
- Very active research area!