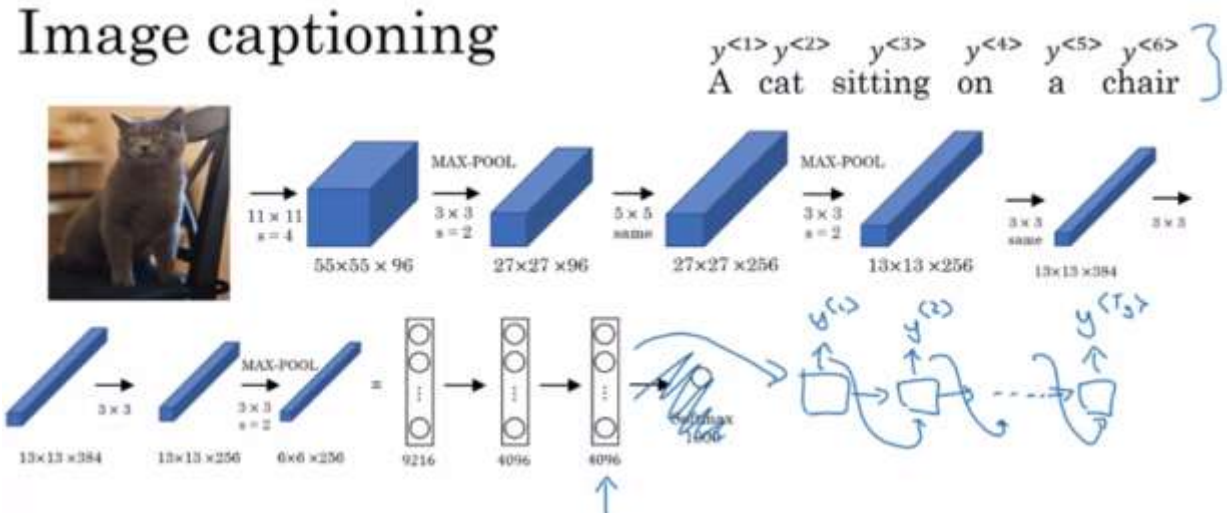
Sequence Models

Week 3



Machine translation objective:

$$\underset{y^{<1>},\ldots,y^{<T_y>}}{\arg\max} \; P(y^{<1>},\ldots,y^{<T_y>} \mid x)$$

In the decoding part, of course we will not use a random probability sampling (like in generating art for example). Greedy search is also not an optimal approach; that is why we will use beam search.

We will use this objective, which has a couple of advantages: (alpha is another hyperparameter)

$$\frac{1}{T_y^{\alpha}} \sum_{t=1}^{T_y} \log P(y^{<t>} \mid x, y^{<1>},\ldots,y^{<t-1>}) \qquad \alpha = 0.7$$

Error analysis on beam search

Human: Jane visits Africa in September. (y^*)

Algorithm: Jane visited Africa last September. ($\hat{y}$)

Case 1: $P(y^*|x) > P(\hat{y}|x)$ ←

Beam search chose $\hat{y}$. But y^* attains higher $P(y|x)$.

Conclusion: Beam search is at fault.

Case 2: $P(y^*|x) \leq P(\hat{y}|x)$ ←

y^* is a better translation than $\hat{y}$. But RNN predicted $P(y^*|x) < P(\hat{y}|x)$.

Conclusion: RNN model is at fault.

Figures out what faction of errors are "due to" beam search vs. RNN model

Bleu score = Bilingual Evaluation Understudy

[Papineni et. al., 2002. Bleu: A method for automatic evaluation of machine translation]

n-gram

$$p_n = \frac{\sum_{n\text{-gram} \in \hat{y}} \text{Count}_{clip}(n\text{-gram})}{\sum_{n\text{-gram} \in \hat{y}} \text{Count}(n\text{-gram})}$$

Bleu details

p_n = Bleu score on n-grams only

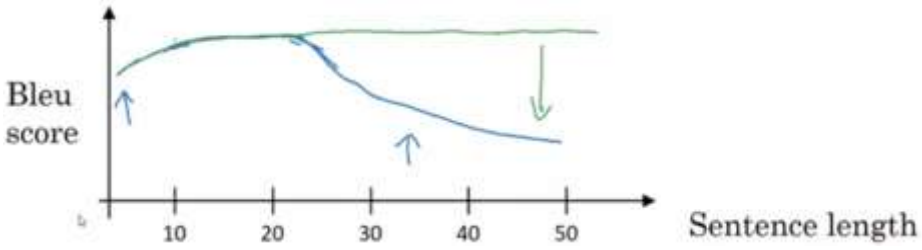
Combined Bleu score: $BP \exp\left(\frac{1}{4}\sum_{n=1}^{4} p_n\right)$

BP = brevity penalty

$$BP = \begin{cases} 1 & \text{if MT\_output\_length} > \text{reference\_output\_length} \\ \exp(1 - \text{MT\_output\_length}/\text{reference\_output\_length}) & \text{otherwise} \end{cases}$$

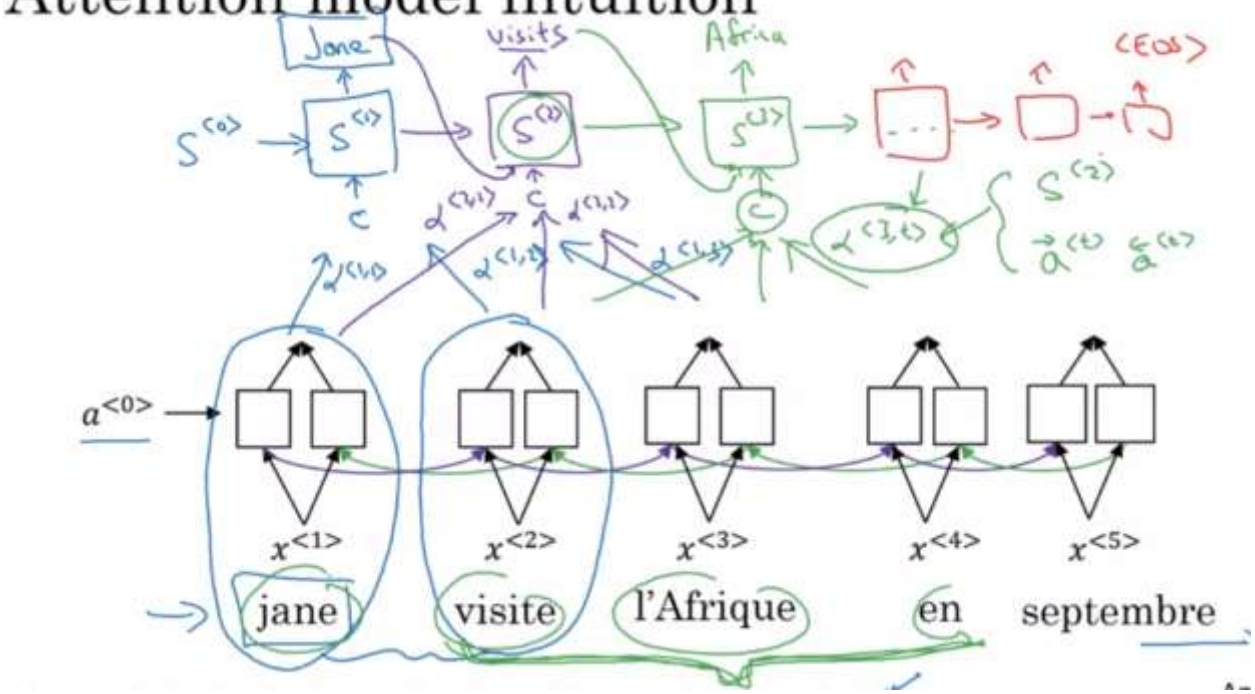
exp(1 − reference_output_length/MT_output_length)

The problem of long sequences (and why we use the Attention Model):

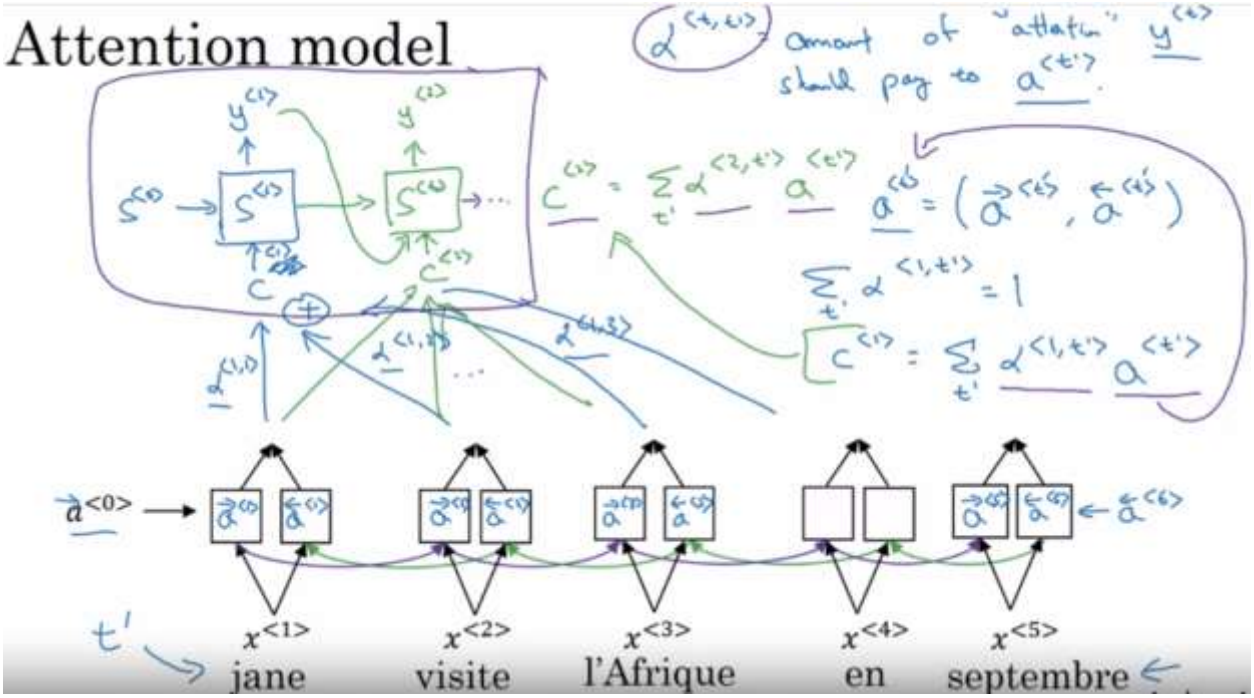


Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate]

Attention model intuition



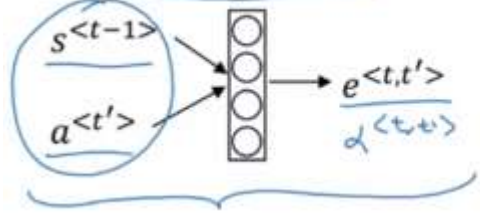
Attention model

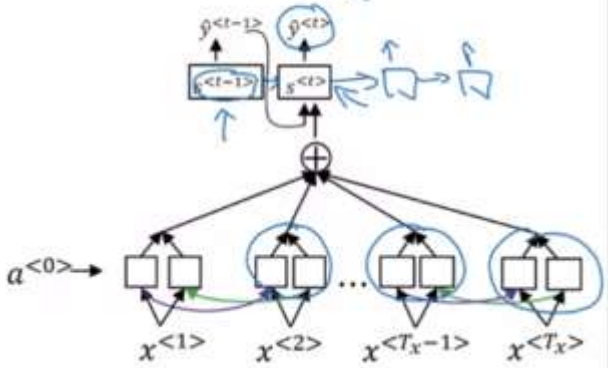


Computing attention $\alpha^{<t,t'>}$

$\alpha^{<t,t'>}$ = amount of attention $y^{<t>}$ should pay to $a^{<t'>}$

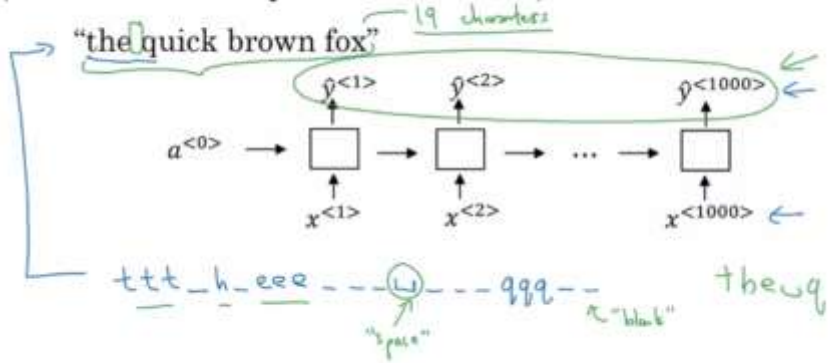
$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$





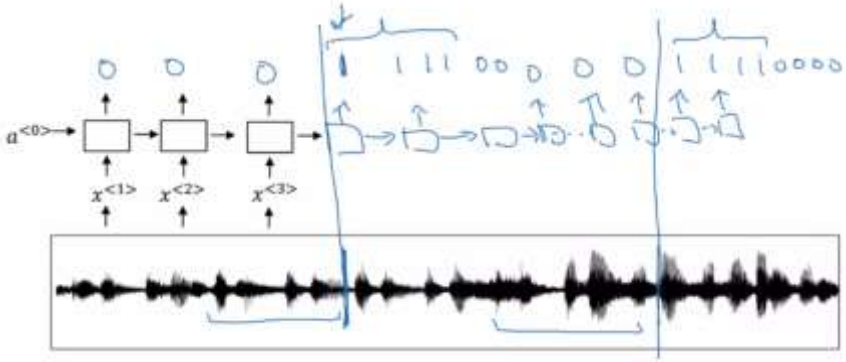
CTC cost for speech recognition

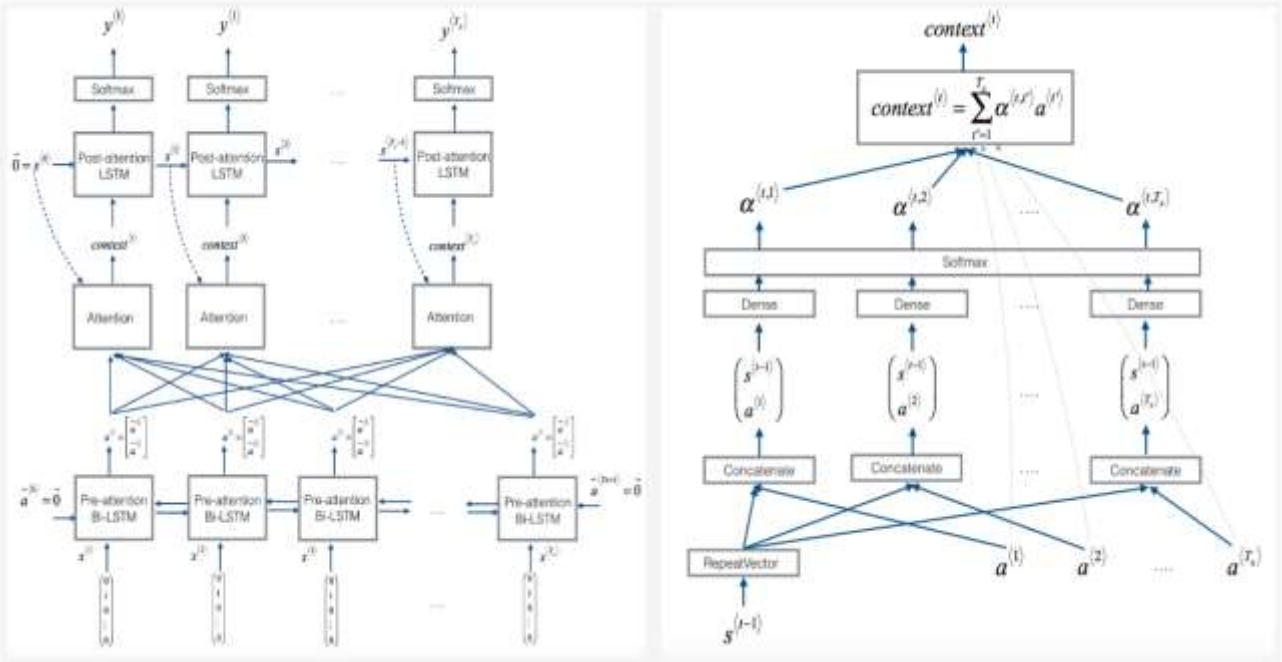
(Connectionist temporal classification)



Basic rule: collapse repeated characters not separated by "blank"

Trigger word detection algorithm





- $s^{\langle t-1 \rangle}$ and $a^{\langle t \rangle}$ are fed into a simple neural network, which learns the function to output $e^{\langle t,t' \rangle}$.
- $e^{\langle t,t' \rangle}$ is then used when computing the attention $a^{\langle t,t' \rangle}$ that $y^{\langle t \rangle}$ should pay to $a^{\langle t' \rangle}$.

What really is an audio recording?

- A microphone records little variations in air pressure over time, and it is these little variations in air pressure that your ear also perceives as sound.
- You can think of an audio recording is a long list of numbers measuring the little air pressure changes detected by the microphone.
- We will use audio sampled at 44100 Hz (or 44100 Hertz).
  - This means the microphone gives us 44,100 numbers per second.
  - Thus, a 10 second audio clip is represented by 441,000 numbers ($= 10 \times 44,100$).

Spectrogram

- It is quite difficult to figure out from this "raw" representation of audio whether the word "activate" was said.
- In order to help your sequence model more easily learn to detect trigger words, we will compute a *spectrogram* of the audio.
- The spectrogram tells us how much different frequencies are present in an audio clip at any moment in time.
- If you've ever taken an advanced class on signal processing or on Fourier transforms:

Instead, it is easier to record lots of positives and negative words, and record background noise separately (or download background noise from free online sources).

Synthesized data is easier to label. In contrast, if you have 10sec of audio recorded on a microphone, it's quite time consuming for a person to listen to it and mark manually exactly when "activate" finished.

- Data synthesis is an effective way to create a large training set for speech problems, specifically trigger word detection.
- Using a spectrogram and optionally a 1D conv layer is a common pre-processing step prior to passing audio data to an RNN, GRU or LSTM.
- An end-to-end deep learning approach can be used to build a very effective trigger word detection system.