Various Sequence to Sequence Architectures

Basic Models : Sequence to Sequence

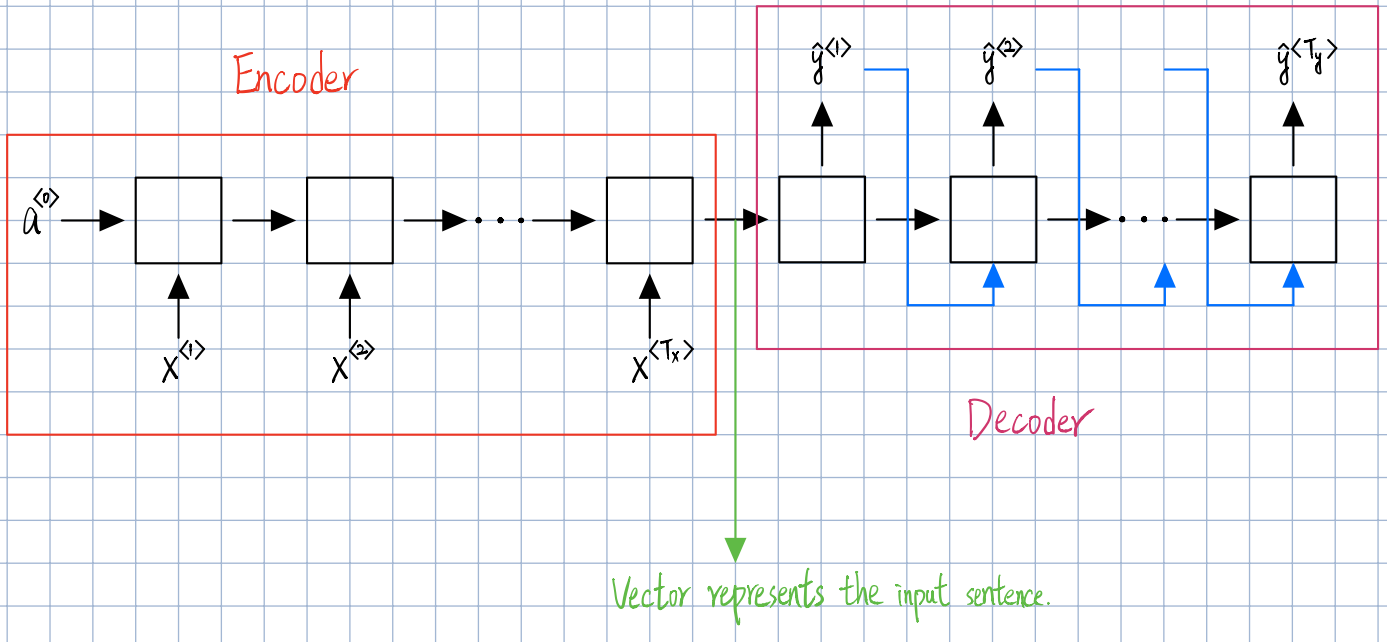
French:

Input: $x^{\langle 1\rangle}$ Jane $x^{\langle 2\rangle}$ visite $x^{\langle 3\rangle}$ l'Afrique $x^{\langle 4\rangle}$ en $x^{\langle 5\rangle}$ septembre.

Task: French to English translation

English:

Output: $y^{\langle 1\rangle}$ Jane $y^{\langle 2\rangle}$ is $y^{\langle 3\rangle}$ visiting $y^{\langle 4\rangle}$ Africa $y^{\langle 5\rangle}$ in $y^{\langle 6\rangle}$ September.



Read: 2014 "Sequence to Sequence Learning with Neural Networks."

2014 "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation"

Image captioning

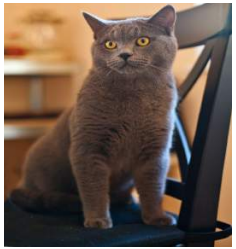
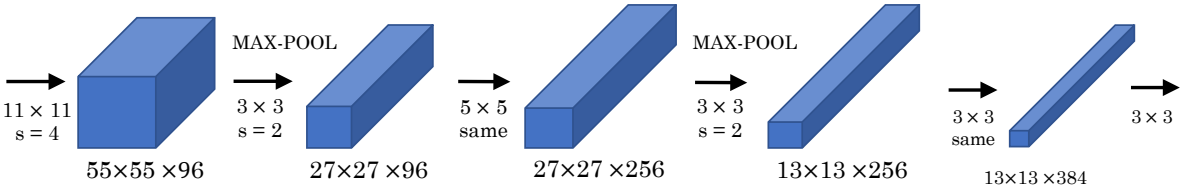
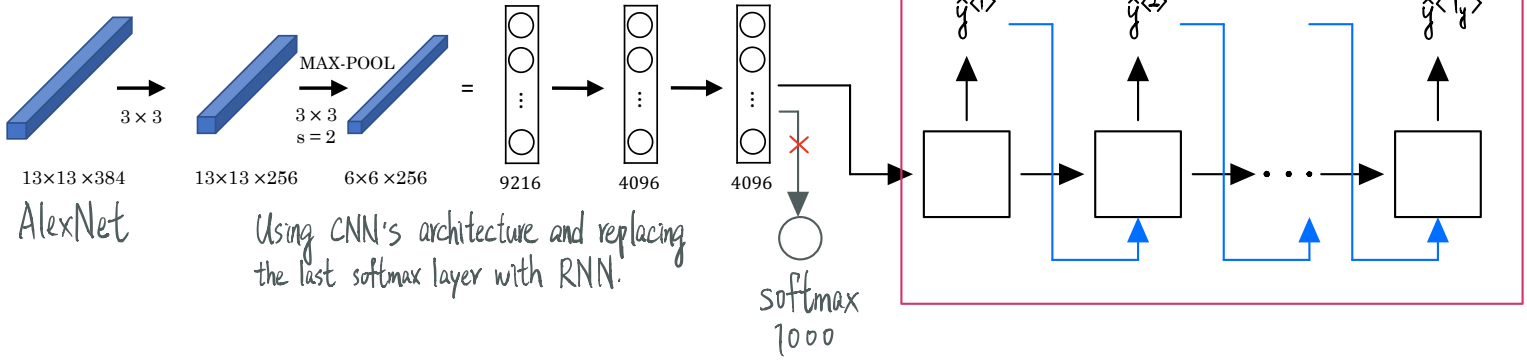


Image to Sequence model

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$

A cat sitting on a chair





AlexNet

Using CNN's architecture and replacing the last softmax layer with RNN.

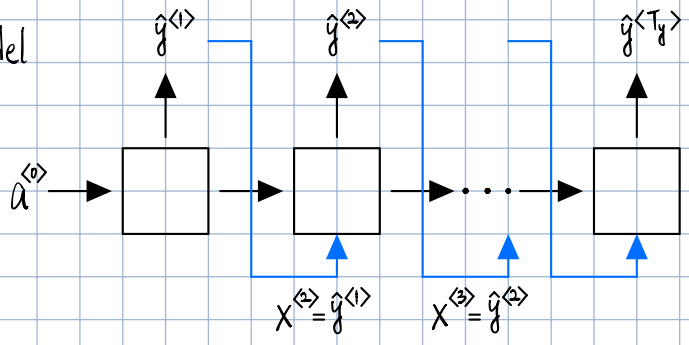
[Mao et. al., 2014. Deep captioning with multimodal recurrent neural networks]
[Vinyals et. al., 2014. Show and tell: Neural image caption generator]
[Karpathy and Li, 2015. Deep visual-semantic alignments for generating image descriptions]

Andrew Ng

Picking the most likely sentence

Previously, we built a language model as follows:

Language Model

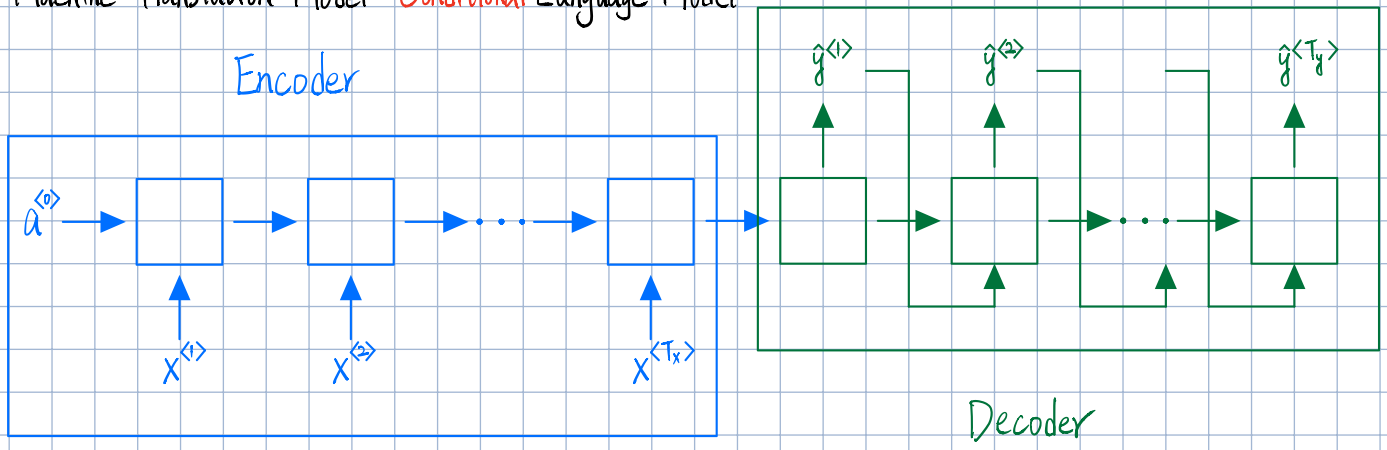


This model estimates the probability of a sentence:

$P(\hat{y}^{\langle 1\rangle}, \hat{y}^{\langle 2\rangle}, \ldots \hat{y}^{\langle T_y\rangle})$

Machine Translation as building a conditional language model

Machine Translation Model: Conditional Language Model



This model estimates the propability of the English output sentence, conditioning on an French input sentence.

$$P(\hat{y}^{\langle 1\rangle}, \hat{y}^{\langle 2\rangle}, \ldots, \hat{y}^{\langle T_y\rangle} \mid x^{\langle 1\rangle}, x^{\langle 2\rangle}, \ldots, x^{\langle T_x\rangle}) = P(\underbrace{\hat{y}^{\langle 1\rangle}, \hat{y}^{\langle 2\rangle}, \ldots, \hat{y}^{\langle T_y\rangle}}_{\text{English}} \mid \overbrace{x}^{\text{French}})$$

Now, how do we find the most likely translation?

Input: Jane visite l'Afrique en septembre.

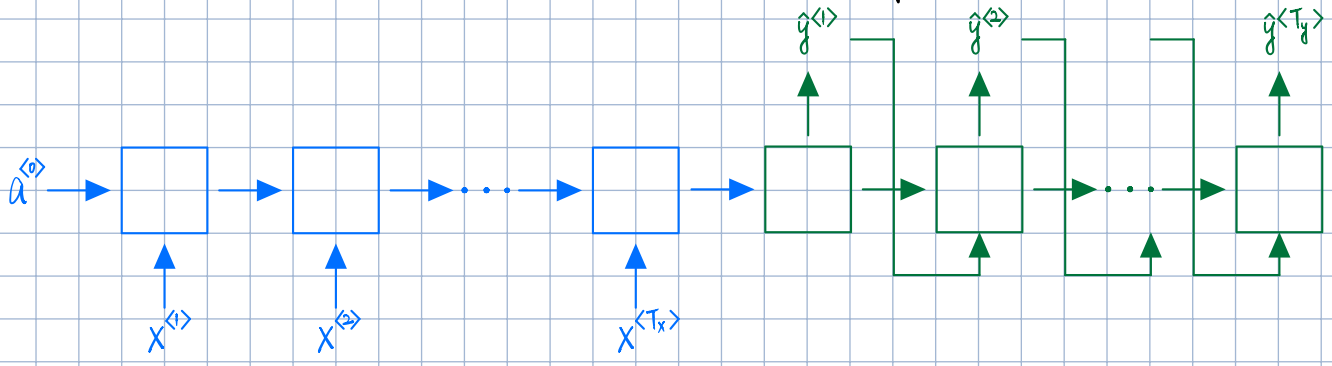
Possible Outputs:

- Jane is visiting Africa in September.
- Jane is going to be visiting Africa in September.

$$\underset{\hat{y}^{\langle 1\rangle}, \ldots, \hat{y}^{\langle T_y\rangle}}{\operatorname{argmax}} P(\hat{y}^{\langle 1\rangle}, \hat{y}^{\langle 2\rangle}, \ldots, \hat{y}^{\langle T_y\rangle} \mid x)$$

Why not a greedy search to find the most likely translation?

A greedy search is first calculating $P(\hat{y}^{\langle 1 \rangle}|X)$, then calculating $P(\hat{y}^{\langle 2 \rangle}|X)$, and so on.



But what we want is the joint possibility: $\underset{\hat{y}^{\langle 1 \rangle}, \ldots, \hat{y}^{\langle T_y \rangle}}{\text{argmax}} P(\hat{y}^{\langle 1 \rangle}, \hat{y}^{\langle 2 \rangle}, \ldots, \hat{y}^{\langle T_y \rangle} | X)$

Here is an example to illustrate:

Possible Outputs:

— Jane is visiting Africa in September. ✓ The better translation.

— Jane is going to be visiting Africa in September. ✗ but this is chosen.

$P(\text{Jane is going}|X) > P(\text{Jane is visiting}|X)$

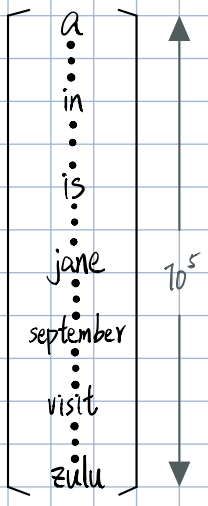
Also, the possible sentences are $(\#\text{vocabulary})^{\#\text{words in sentence}}$

This problem can be solved by Beam Search algorithm.

Beam Search

Beam search is a heuristic search algorithm that explores a graph by expanding the most promising node in a limited set.

Dictionary =



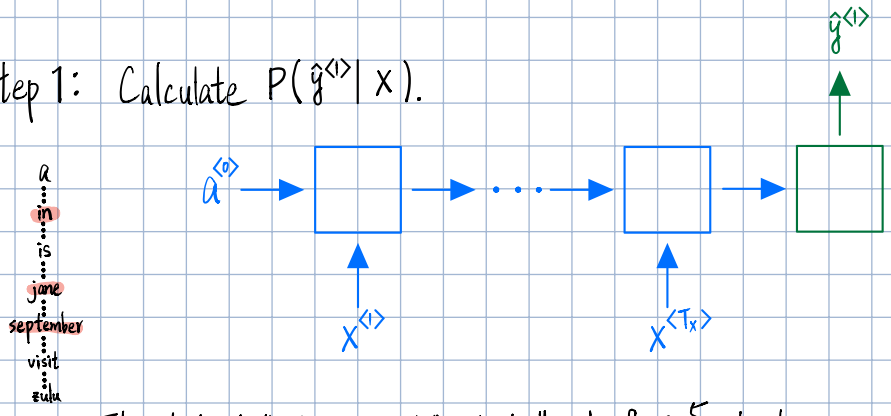
Given an input sentence and a dictionary with size = 10^5

Input: Jane visite l'Afrique en septembre.

beam width = 3. Number of the most possible words to consider in each step.

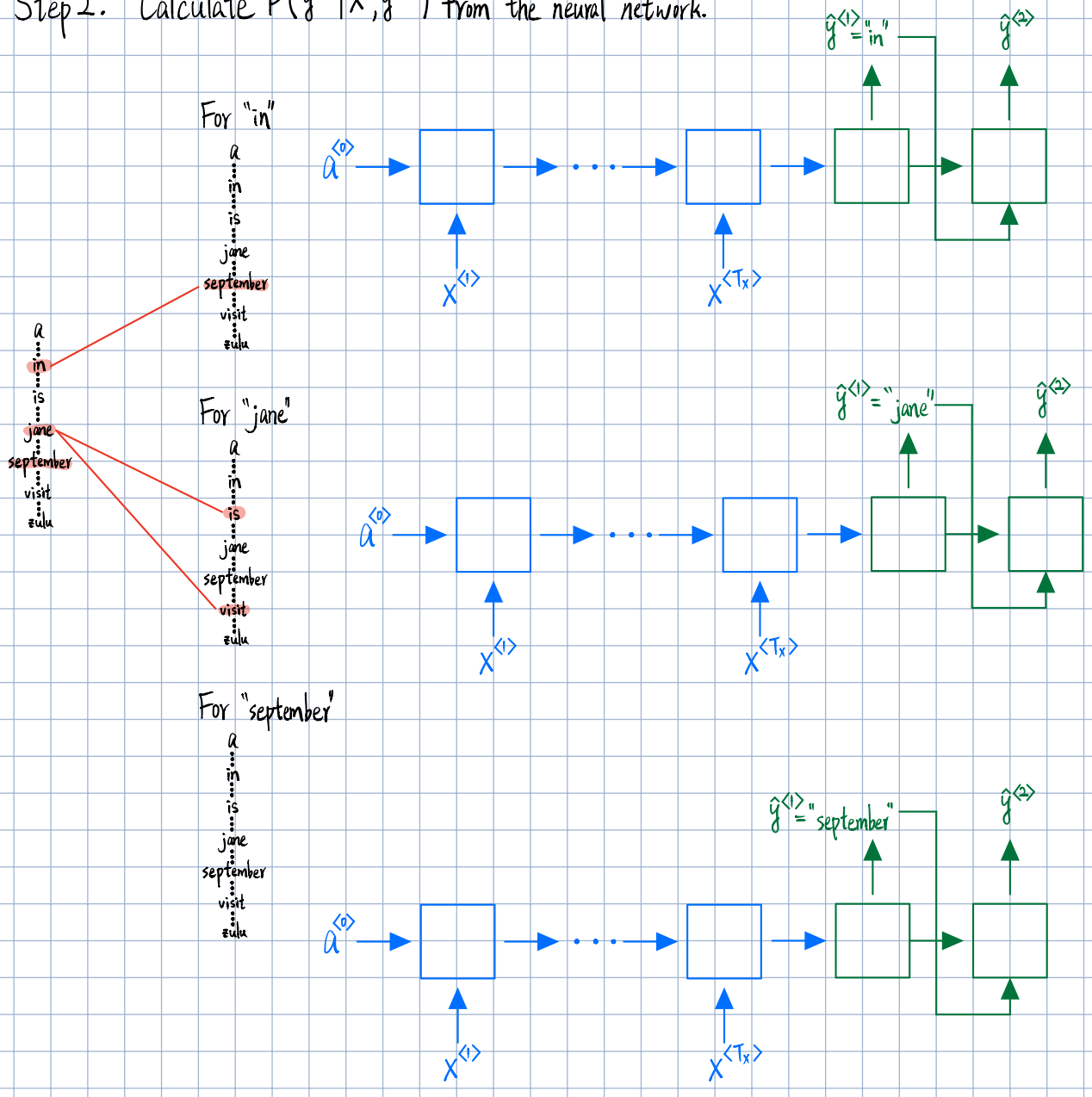
If beam width = 1, then it's exactly greedy search ?

Step 1: Calculate $P(\hat{y}^{<1>} | x)$.



It selects "in", "jane", and "september" out of 10^5 outputs.

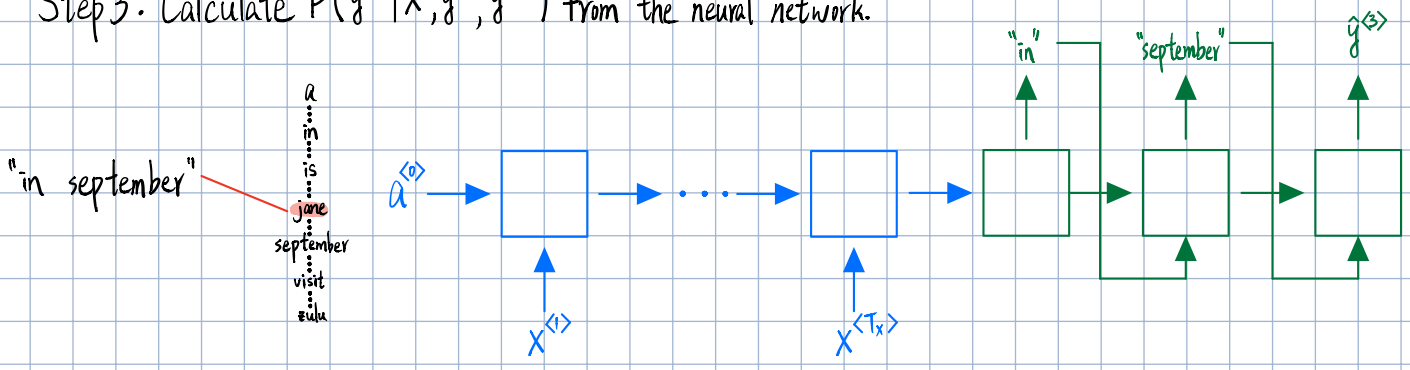
Step 2: Calculate $P(\hat{y}^{<2>} | x, \hat{y}^{<1>})$ from the neural network.

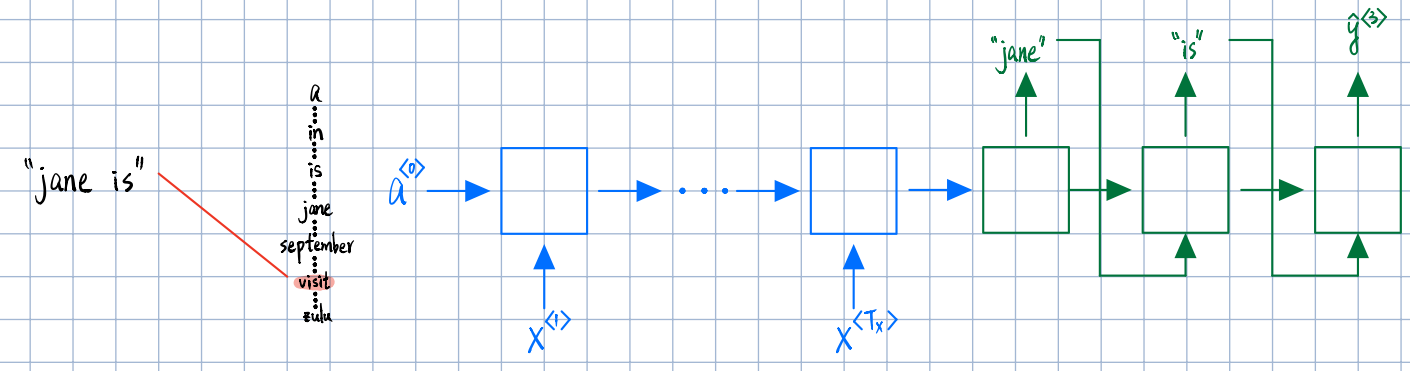


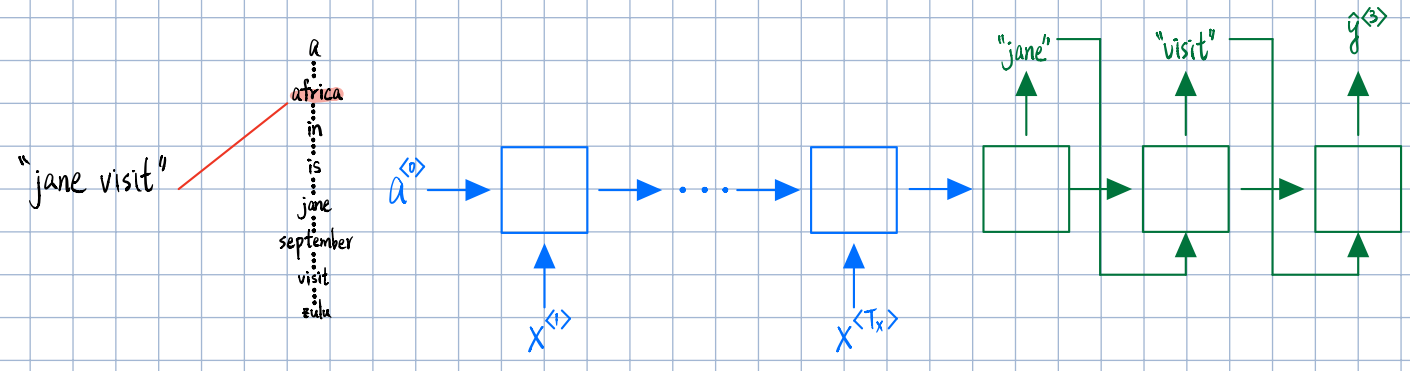
What we want is $P(\hat{y}^{<1>}, \hat{y}^{<2>} | x) = \boxed{P(\hat{y}^{<1>} | x)}\, P(\hat{y}^{<2>} | x, \hat{y}^{<1>})$ (from step 1.)

There are $3 \cdot 10^4$ choices in step 2. We choose top 3 $P(\hat{y}^{<1>}, \hat{y}^{<2>} | x)$ out of them.

Step 3: Calculate $P(\hat{y}^{\langle 3 \rangle} | x, \hat{y}^{\langle 1 \rangle}, \hat{y}^{\langle 2 \rangle})$ from the neural network.







Again, we pick 3 out of 3×10^4 possibilities.

$$P(\hat{y}^{\langle 1 \rangle}, \hat{y}^{\langle 2 \rangle}, \hat{y}^{\langle 3 \rangle} | x) = \overbrace{\boxed{P(\hat{y}^{\langle 1 \rangle}, \hat{y}^{\langle 2 \rangle} | x)}}^{\text{from step 2}} P(\hat{y}^{\langle 3 \rangle} | x, \hat{y}^{\langle 1 \rangle}, \hat{y}^{\langle 2 \rangle})$$

$$= P(\hat{y}^{\langle 1 \rangle} | x) \, P(\hat{y}^{\langle 2 \rangle} | x, \hat{y}^{\langle 1 \rangle}) \, P(\hat{y}^{\langle 3 \rangle} | x, \hat{y}^{\langle 1 \rangle}, \hat{y}^{\langle 2 \rangle})$$

Refinements to Beam Search

$$P(\hat{y}^{<1>}, \hat{y}^{<2>}, \ldots, \hat{y}^{<T_y>} \mid x)$$

$$= P(\hat{y}^{<1>} \mid x)\, P(\hat{y}^{<2>} \mid x, \hat{y}^{<1>})\, P(\hat{y}^{<3>} \mid x, \hat{y}^{<1>}, \hat{y}^{<2>}) \cdots P(\hat{y}^{<T_y>} \mid x, \hat{y}^{<1>}, \ldots, \hat{y}^{<T_y-1>})$$

$$= \prod_{t=1}^{T_y} P(\hat{y}^{<t>} \mid x, \hat{y}^{<1>}, \ldots, \hat{y}^{<t-1>})$$

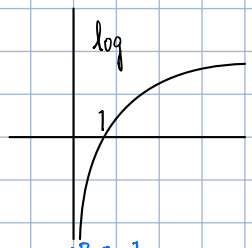
Length Normalization

$$\underset{\hat{y}}{\text{argmax}} \prod_{t=1}^{T_y} P(\hat{y}^{<t>} \mid x, \hat{y}^{<1>}, \ldots, \hat{y}^{<t-1>})$$ This equation is numerical instable due to probality being float.

$$\rightarrow \underset{\hat{y}}{\text{argmax}} \sum_{t=1}^{T_y} \log P(\hat{y}^{<t>} \mid x, \hat{y}^{<1>}, \ldots, \hat{y}^{<t-1>}) \not\equiv \underset{y}{\text{argmax}} \sum_{t=1}^{T_y} P(\hat{y}^{<t>} \mid x, \hat{y}^{<1>}, \ldots, \hat{y}^{<t-1>})$$

This model tends to make the output sentence shorter. More output, more negative numbers are added together, and smaller the model.

Why? $\because \log(P) < 0$



$\because P: 0 \sim 1$

$\therefore \log(P) < 0$

Thus, we can normalize this function by:

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

$\alpha = 0$, no normalization

$\alpha = 1$, normalize by the length of the output sentence

$\alpha = 0.7$, heuristic value

Discussion:

Large beam width: better result, more memory, running slower.

Small beam width: worse result, less memory, running faster.

Beam width = 10 (Product) 100 1000+ (Publish papers)

Unlike exact search algorithms like BFS (Breadth First Search) or DFS (Depth First Search), Beam Search runs faster but is not guaranteed to find exact maximum for $\underset{\hat{y}}{\text{argmax}} P(\hat{y} \mid x)$.

Error Analysis in Beam Search

At this point, we can use RNN models and beam search to run machine translation.

But if the translation result is bad, how do I know whom to blame. RNN models or beam search or me.

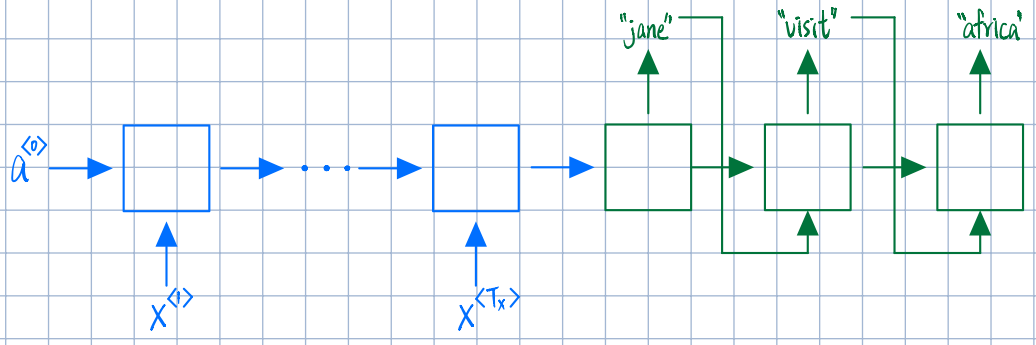
Eg:

French: Jane visite l'Afrique en septembre.

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

Algorithm: Jane visited Africa in September. ($\hat{y}$) A quite bad translation.

I can use the RNN model to compute $P(y^*|x)$ and $P(\hat{y}|x)$, and see which one is larger.



Case 1: $P(y^*|x) > P(\hat{y}|x)$

Conclusion: Beam search is to be blamed.

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

Conclusion: RNN model is to be blamed.

Now, acquire more samples and go through this process and the sinner will be found.

Bleu Score

Read "BLEU: a Method for Automatic Evaluation of Machine Translation"

One of the challenges of machine translation is that, given a French sentence, there could be multiple English translations that are equally good translations of that French sentence. How do I evaluate a machine translation system if there are multiple equally good answers.

BiLingual Evaluation Understudy (BLEU)

Given French: Le chat est sur le tapis.

Reference 1: The cat is on the mat. 2 "the"

Reference 2: There is a cat on the mat. 1 "the"

2 > 1 ∴ select 2

MT (Machine Translation) output: the the the the the the the Terrible Translation

(Wrong) Precision : $\frac{7 \text{ Every "the" is in either one of Reference 1 and 2.}}{7 \text{ "the" in output}} = 1$. The score is high but it's a terrible translation. Don't use it.

Modified Precision : $\frac{\text{Each word has only the credit up to the maximum number of times it appears in the reference sentences}}{7 \text{ "the" in output}} = \frac{2}{7}$

BLEU score on bigram: Reference 1: The cat is on the mat.

Reference 2: There is a cat on the mat.

MT output: The cat the cat on the mat.

Bigram	"the cat"	"cat the"	"cat on"	"on the"	"the mat"
Count in MT:	2	1	1	1	1
$\text{Count}_{\text{clip}}$ (max times of occurences in references) :	1	0	1	1	1

Thus, the Precision $= \frac{1+1+1+1}{2+1+1+1+1}$

$= \frac{4}{6}$

BLEU score on unigram: Reference 1: The cat is on the mat.

Reference 2: There is a cat on the mat.

MT output: The cat the cat on the mat. ($\hat{y}$)

Define P_1 as the Precision for unigram.
P_2 as the Precision for bigram.
$\vdots$
P_n as the Precision for n-gram.

$$P_1 = \frac{\sum_{\text{unigram} \in \hat{y}} \text{Count}_{\text{clip}}(\text{unigram})}{\sum_{\text{unigram} \in \hat{y}} \text{Count}(\text{unigram})} \qquad P_1 = \frac{\sum_{\text{n-gram} \in \hat{y}} \text{Count}_{\text{clip}}(\text{n-gram})}{\sum_{\text{n-gram} \in \hat{y}} \text{Count}(\text{n-gram})}$$

Given P_n as BLEU score on n-gram only. $n = 1, 2, 3, 4$

Combine BLEU score : $BP \cdot \exp(\sum_{n=1}^{4} P_n)$ BP : Brevity Penalty

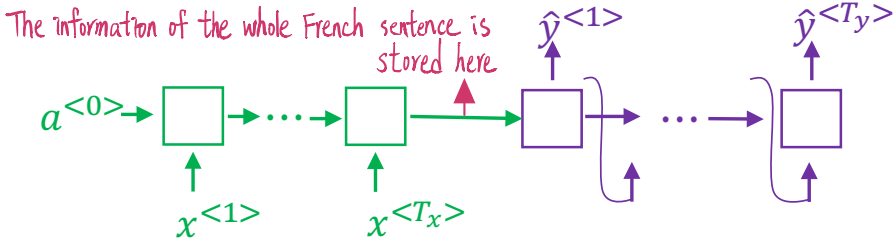
$$BP = \begin{cases} 1 & \text{, if MT\_output\_length} > \text{reference\_output\_length} \\ \exp\left(1 - \frac{\text{reference\_output\_length}}{\text{MT\_output\_length}}\right) & \text{, otherwise} \end{cases}$$

deeplearning.ai

Sequence to sequence models

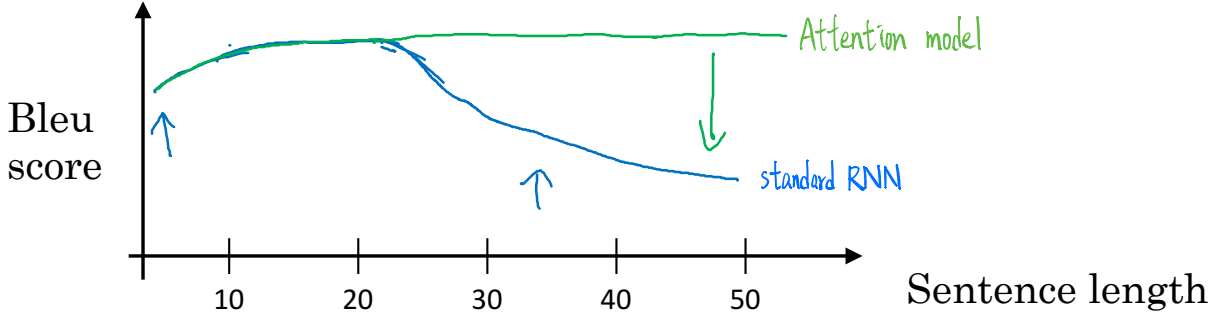
Attention model intuition

The problem of long sequences



Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

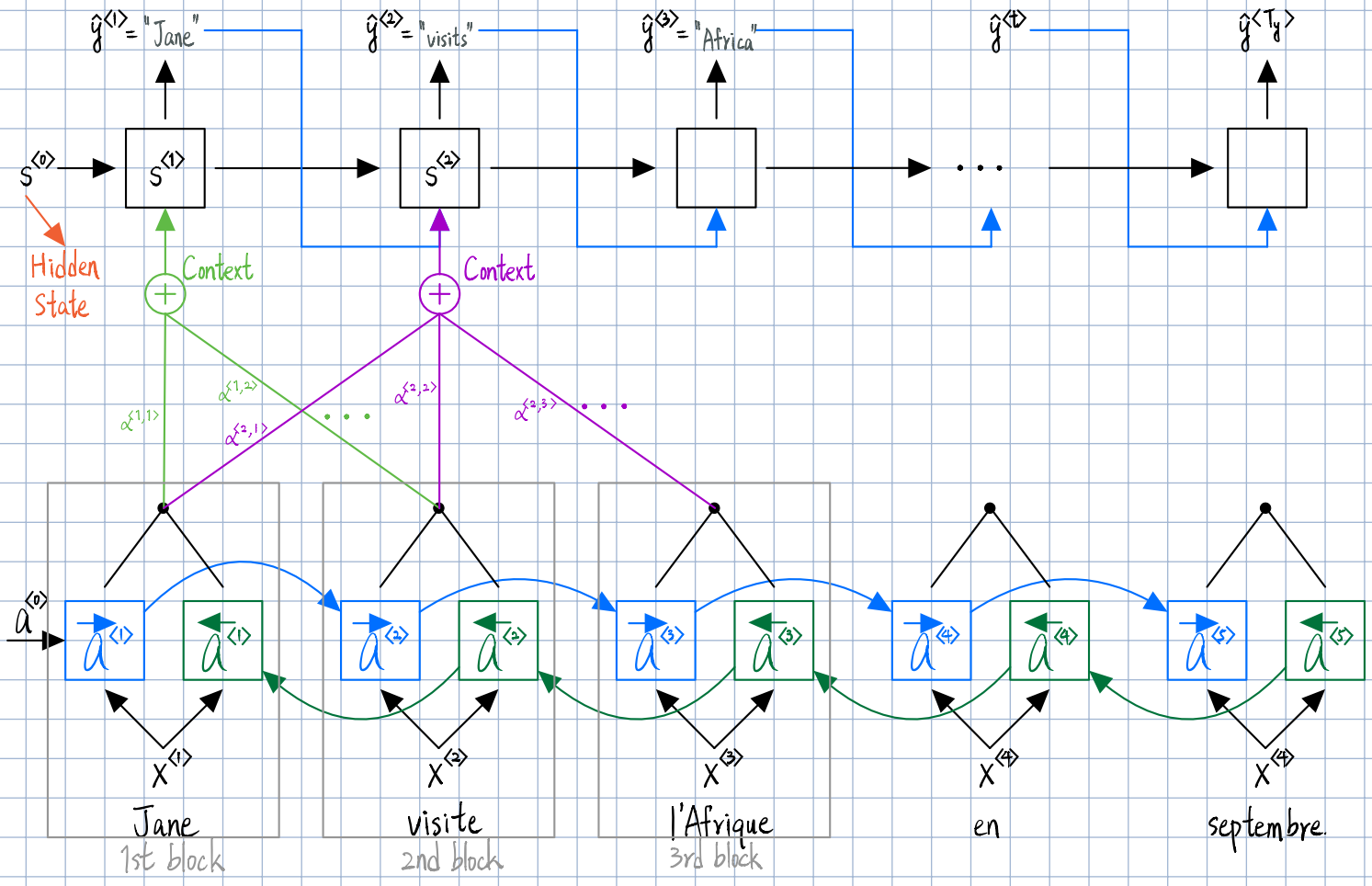
Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.



Andrew Ng

Attention Model Intuition

Read 2014 "Neural Machine Translation by Jointly Learning to Align and Translate"

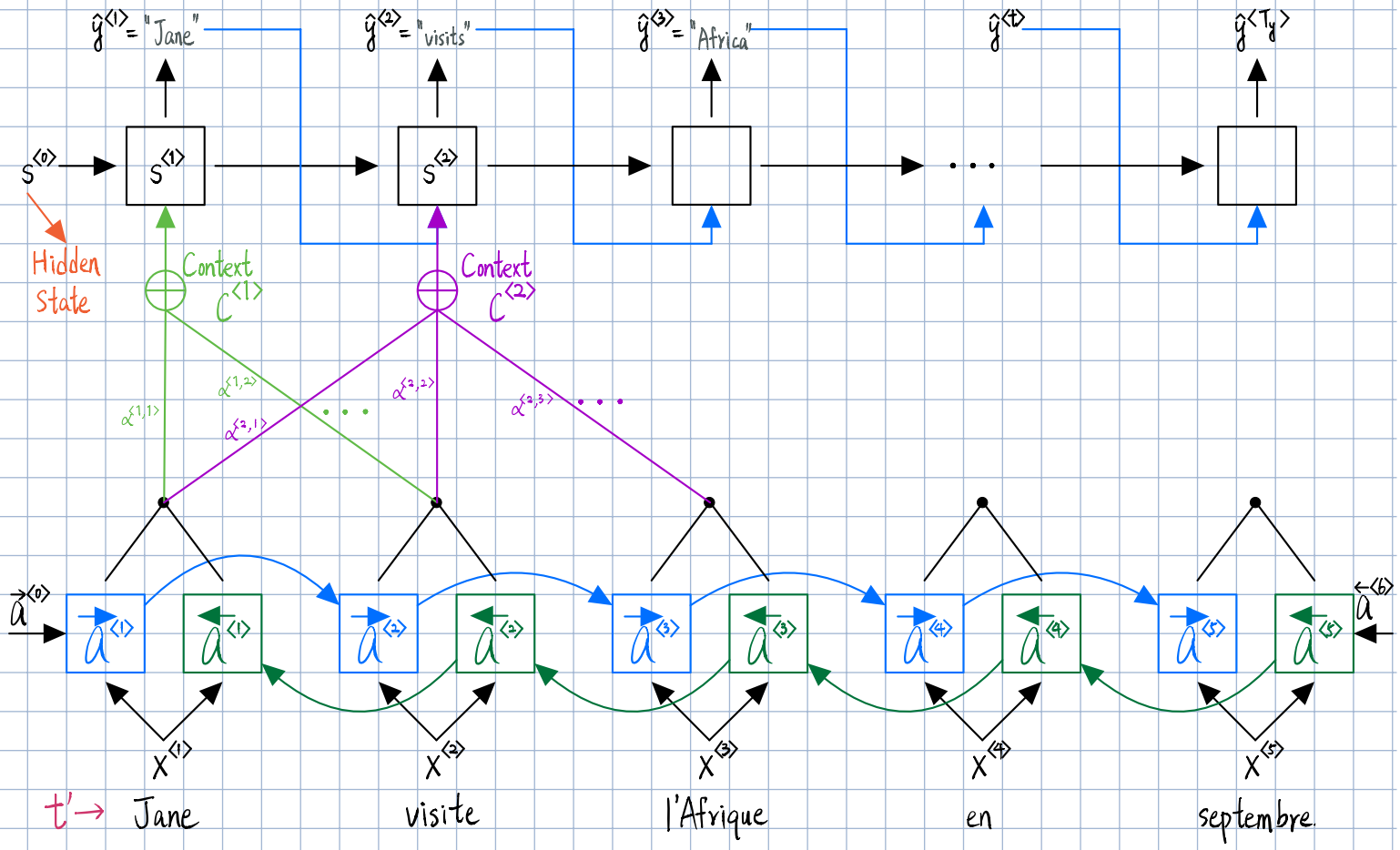


Attention Weight $\alpha^{\langle 1,1 \rangle}$: When the model is generating the first output, how much it should be paying attention to the first block.

$\alpha^{\langle 1,2 \rangle}$: When the model is generating the first output, how much it should be paying attention to the second block.

$\alpha^{\langle 2,1 \rangle}$: When the model is generating the second output, how much it should be paying attention to the first block.

Attention Model Implementation



Redefine the time step of the input sentence as t' to distinguish from the time step t of the output sentence.

Thus, $a^{\langle t' \rangle} = (\overrightarrow{a}^{\langle t' \rangle}, \overleftarrow{a}^{\langle t' \rangle})$

The attention weights for the first output word $\sum_{t'=1}^{T_x} \alpha^{\langle 1,t' \rangle} = 1$

The context C for the first output word $C^{\langle 1 \rangle} = \sum_{t'=1}^{T_x} \alpha^{\langle 1,t' \rangle} a^{\langle t' \rangle}$. For the second output word $C^{\langle 2 \rangle} = \sum_{t'=1}^{T_x} \alpha^{\langle 2,t' \rangle} a^{\langle t' \rangle}$

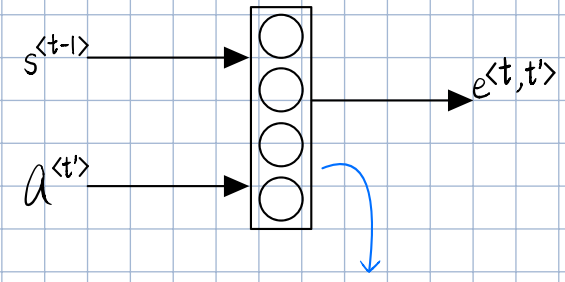
$\alpha^{\langle t,t' \rangle}$: the amount of "Attention" that $\hat{y}^{\langle t \rangle}$ should pay to $a^{\langle t' \rangle}$. In other words, when the model is generating the t-th output word, how much attention it should be paying to the t'-th input word.

Computing Attention $\alpha^{\langle t,t' \rangle}$

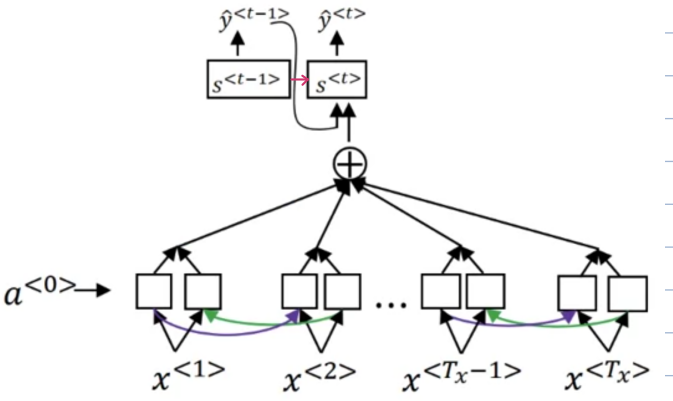
$$\alpha^{\langle t,t' \rangle} = \frac{\exp(e^{\langle t,t' \rangle})}{\sum_{t'=1}^{T_x} \exp(e^{\langle t,t' \rangle})}$$

This is why $\sum_{t'=1}^{T_x} \alpha^{\langle t,t' \rangle} = 1$

Train a small neural network to generate $e^{\langle t,t' \rangle}$.



This neural network tells how much attention $\hat{y}^{\langle t \rangle}$ should pay to $a^{\langle t' \rangle}$.



The intuitive explanation is that $e^{\langle t,t' \rangle}$ is depending on the state of the previous time step $s^{\langle t-1 \rangle}$ and the associated $a^{\langle t' \rangle}$, which makes sense.

$s^{\langle t \rangle}$ depends on $\alpha^{\langle t,t' \rangle}$ which in turn depends on $e^{\langle t,t' \rangle}$.

One downside of attention model is that it takes quadratic cost to run this algorithm. If there are T_x words in the input and T_y words in the output, then the total number of attention parameters $\alpha^{\langle t,t' \rangle}$ is $T_x \cdot T_y$.

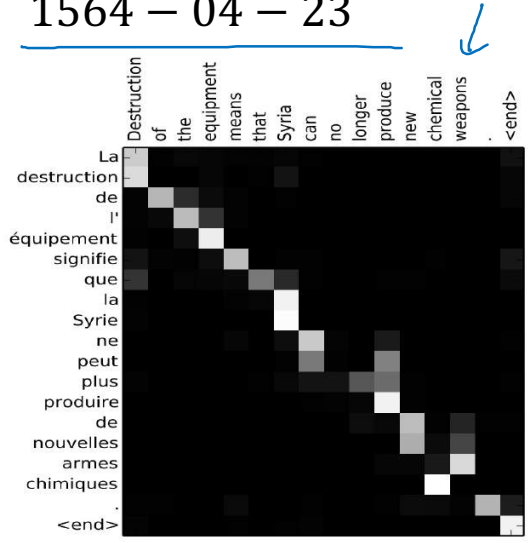
Read 2015 "Show, Attention and Tell: Neural Image Caption Generation with Visual Attention".

Attention examples

July 20th 1969 ⟶ 1969 – 07 – 20

23 April, 1564 ⟶ 1564 – 04 – 23

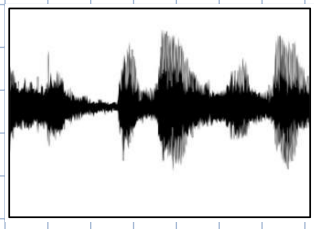
Visualization of $\alpha^{<t,t'>}$:

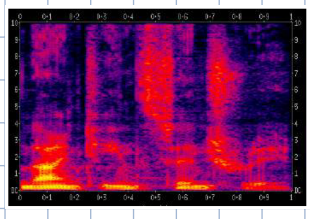


Andrew Ng

Speech Recognition

Audio clip X → Transcript Y





"the quick brown fox"

Once upon a time, speech recognition system is used to be built using Phonemes.

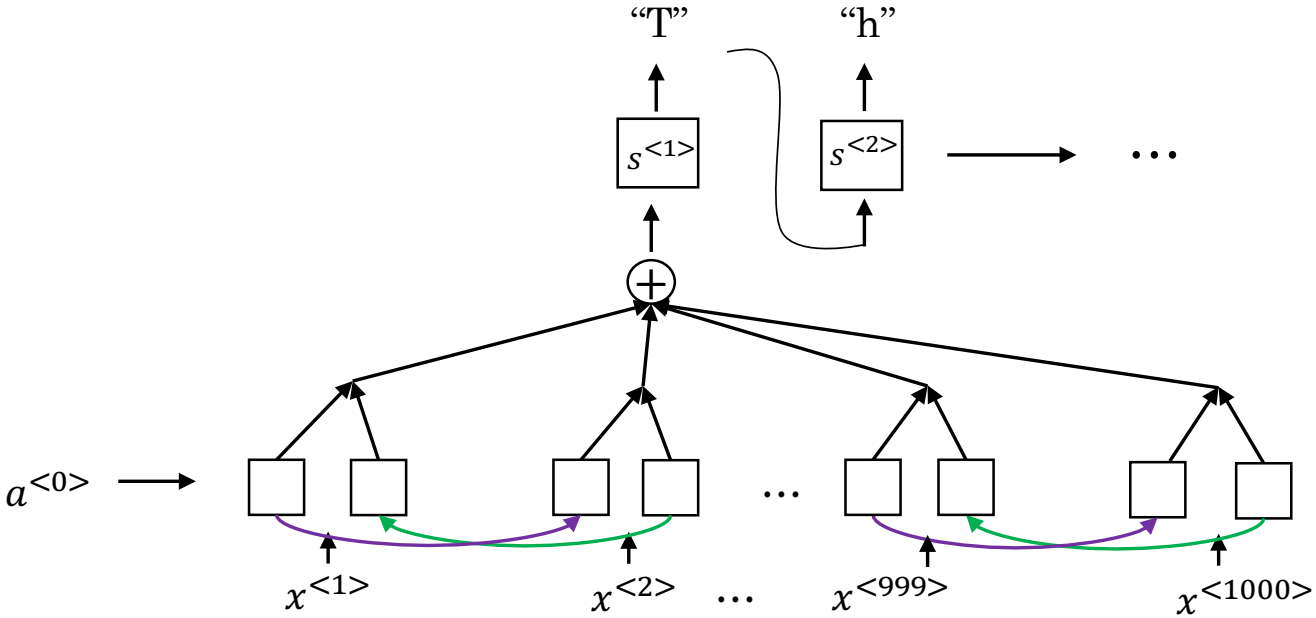
Phonemes are hand-crafted basic units of sound.

People used to decompose audio into phonemes to conduct speech recognition.
But with end-to-end deep learning, people find that phonemes representations are no longer necessary.
We just need More data.

In academia, $300h \sim 3000h$ data.

In commercial, $10^4h \sim 10^5h$ data.

Attention model for speech recognition



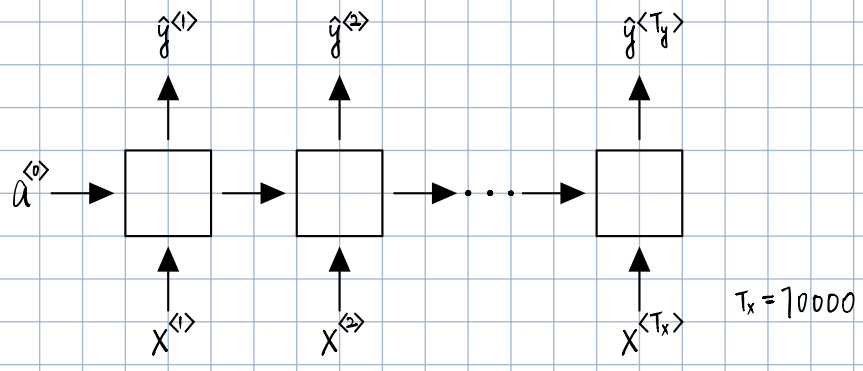
CTC cost for speech recognition

CTC: Connectionist Temporal Classification

Read 2006 "Connectionist Temporal Classification: Labeling Unsegmented Sequence Data with Recurrent Neural Networks"

audio = "the quick brown fox"

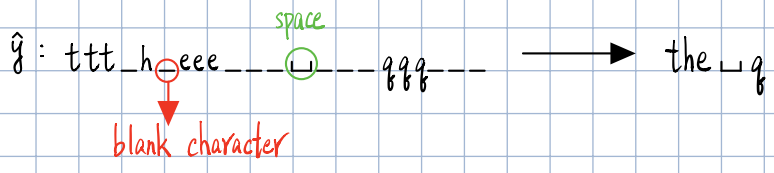
In practice, bidirectional RNN with LSTM or GRU is used.



$T_x = 10000$

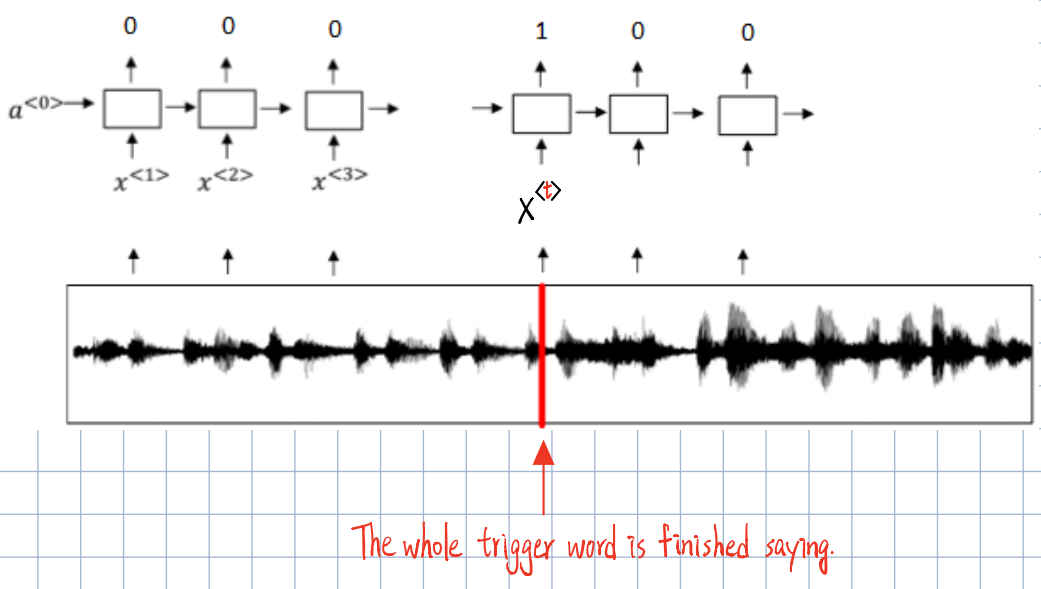
In speech recognition, the number of input time steps is much bigger than the number of output time steps. Eg, 10 seconds audio, sampled with 1k Hz, has 10k input time step, while the output words are only few words.

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



Trigger Word Detection

Eg: Alexa
Hey Siri
Okay Google



When the target label of $X^{<t>}$ is set to 1, it means a trigger word has been said.

$X^{<t>}$ represents the feature of the audio at time t.

However, the above scheme creates a imbalanced training set.

So, we can set the output target label a few 1s for a fixed period of time before resetting to 0.