Sequence models → Week 03 (Attention mechanism)

aakashgoel12.medium.com/sequence-models-week-03-attention-mechanism-111345e5a8c0

February 21, 2021

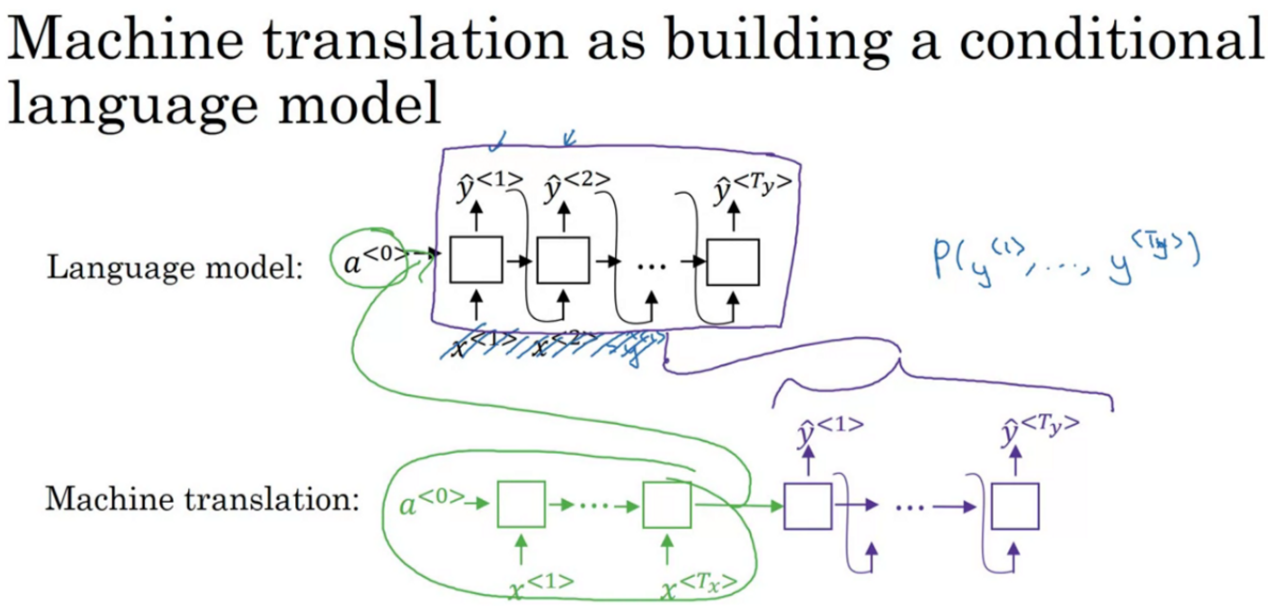
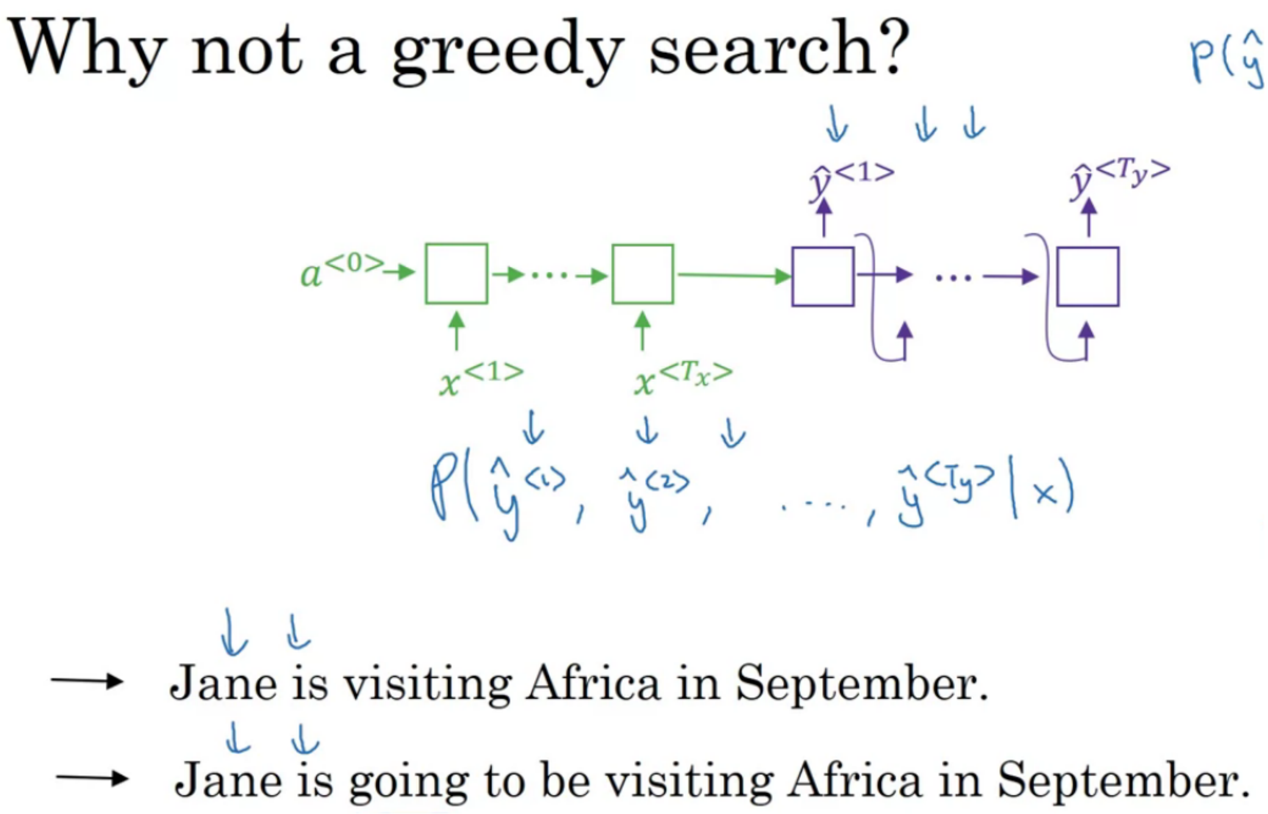


Image 01

In Language Model, we find probability of sentence.

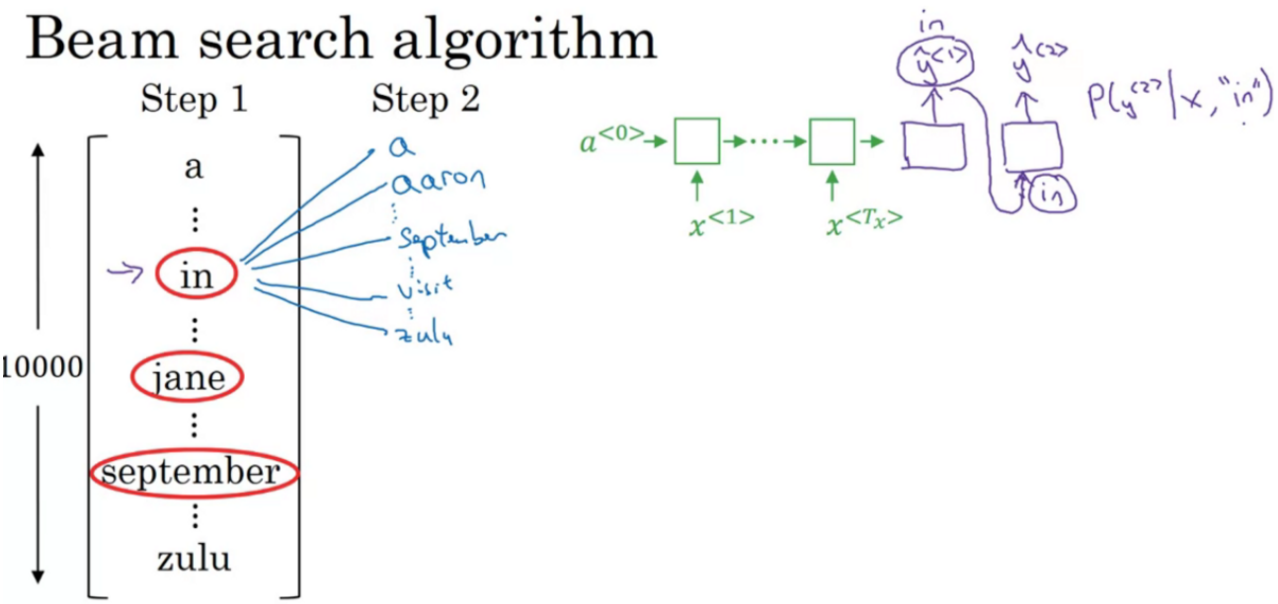
Decoder in Machine Translation system is same as Language Model and a<0> in language model is similar to Encoder in Machine Translation.

In M/C Translation, we use beam search instead of greedy search.



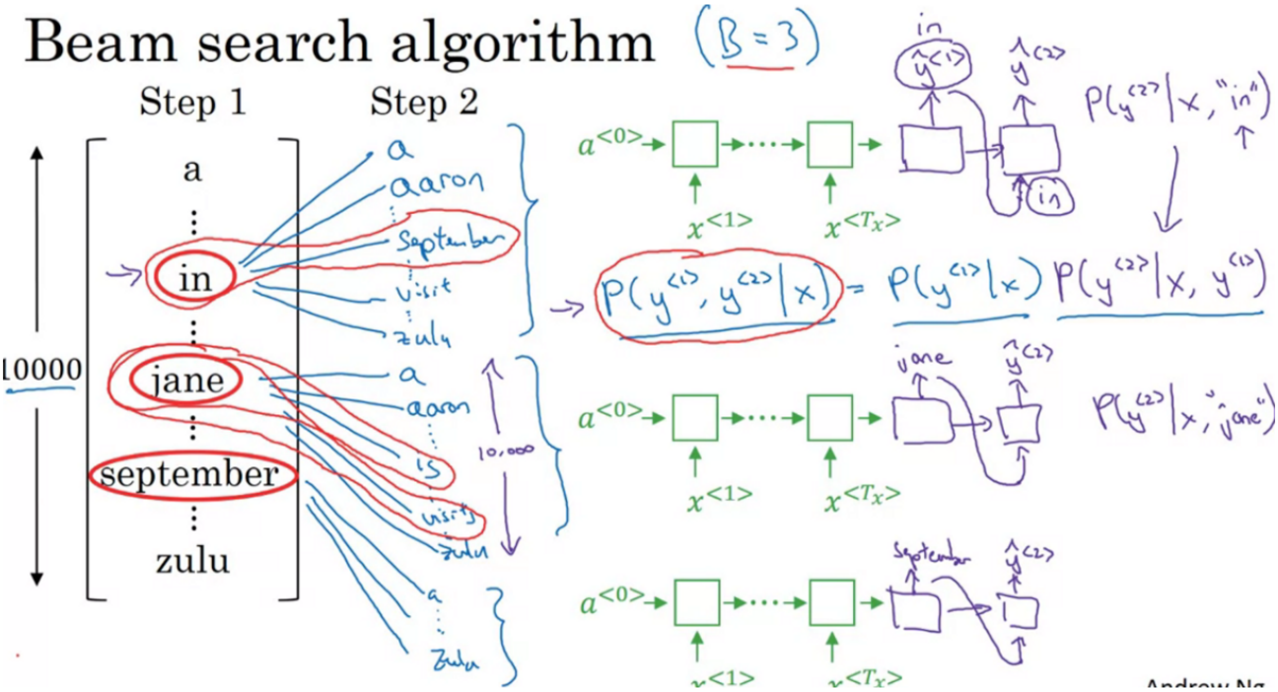
P(Jane is going/X) > P(Jane is visiting/X) but sentence 1 is more optimial

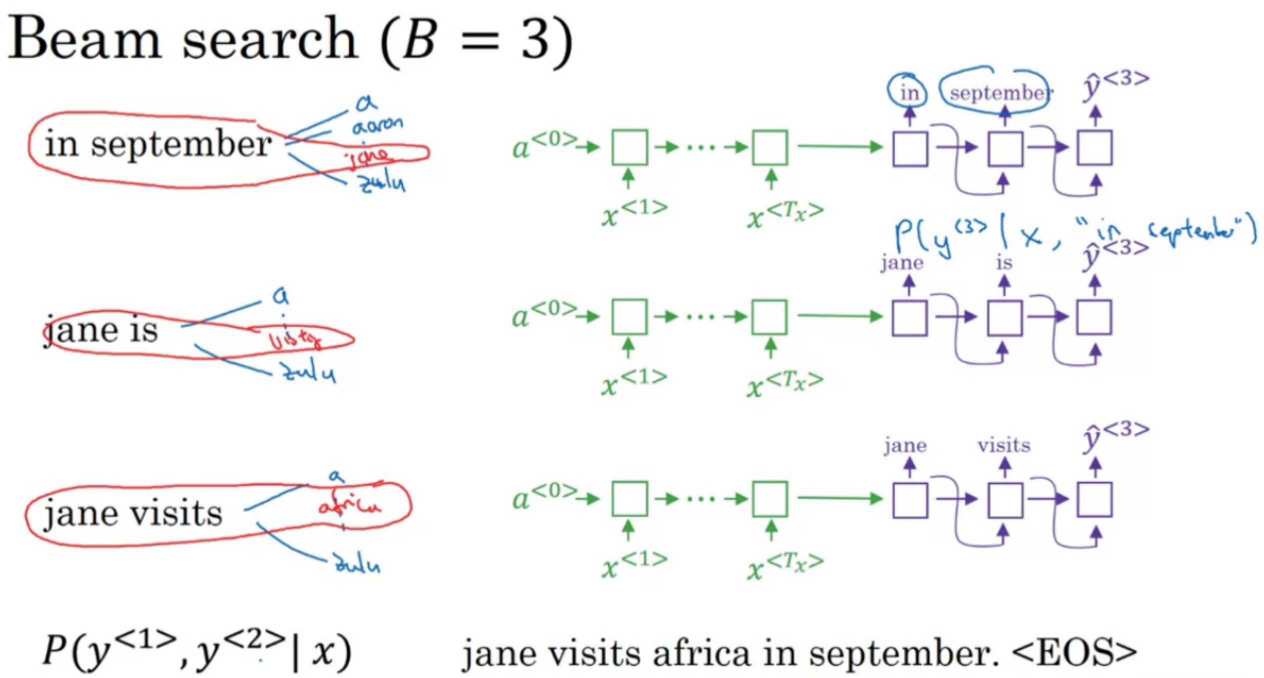
Beam Search



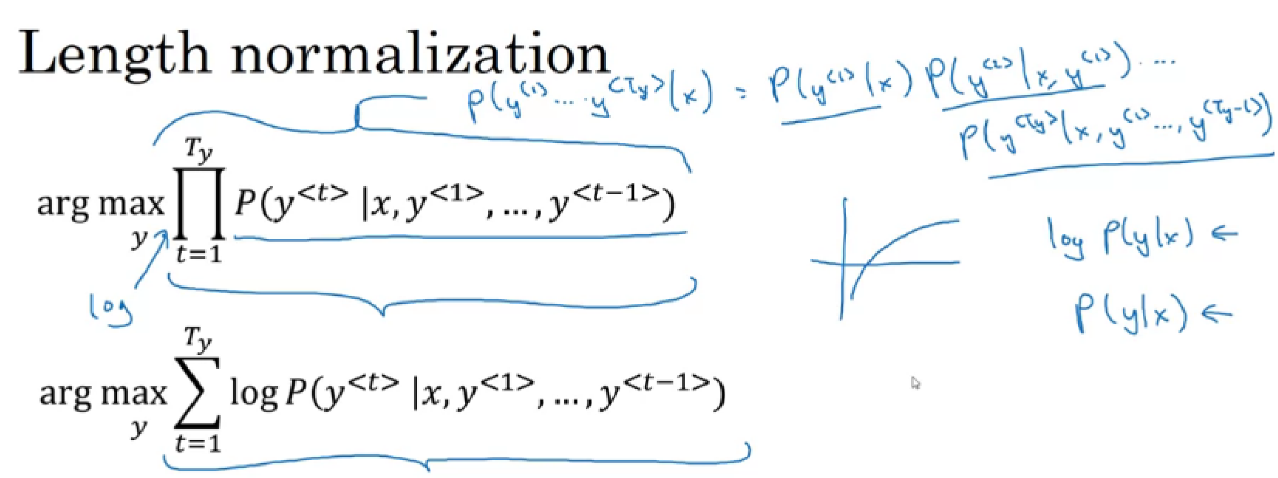
Beam Width considered is 3 i.e. Top 3 words will be considered as candidate..

Say, Word1= “in”, need to find P(Y2/X,”in”) i.e. Prob. of Y2 given X and “in”.



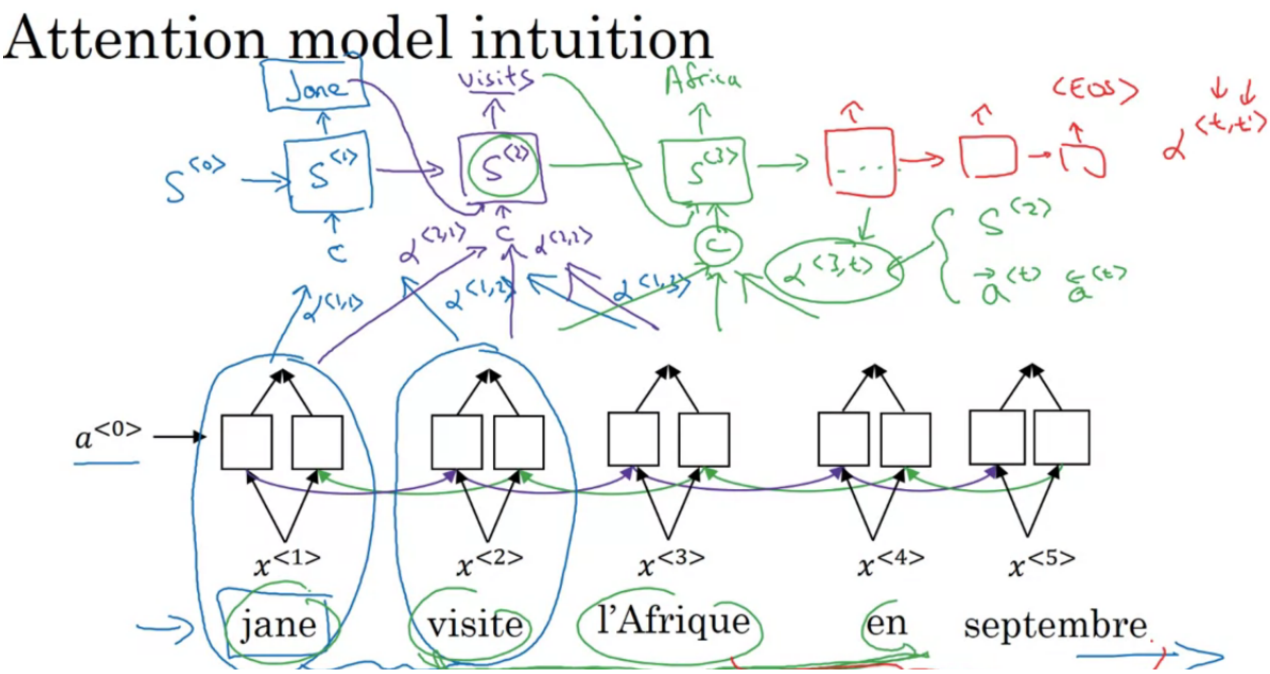


Log is strictly **monotonically increasing** function i.e. maximizing P(Y/X) is same as maximizing Log (p(Y/X)) ..

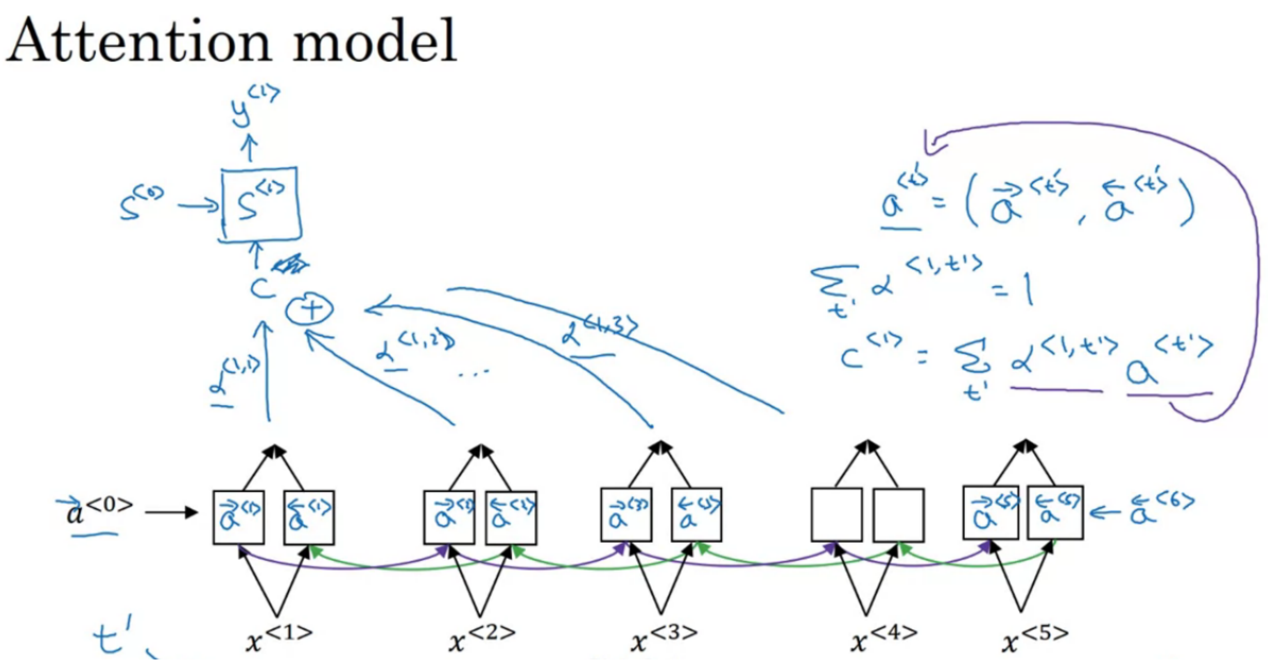


Above P(Yt/X,Y1,……,Yt-1), Unnaturally tends/prefer short translations as multiplying no less than 1 will give short tiny number ..

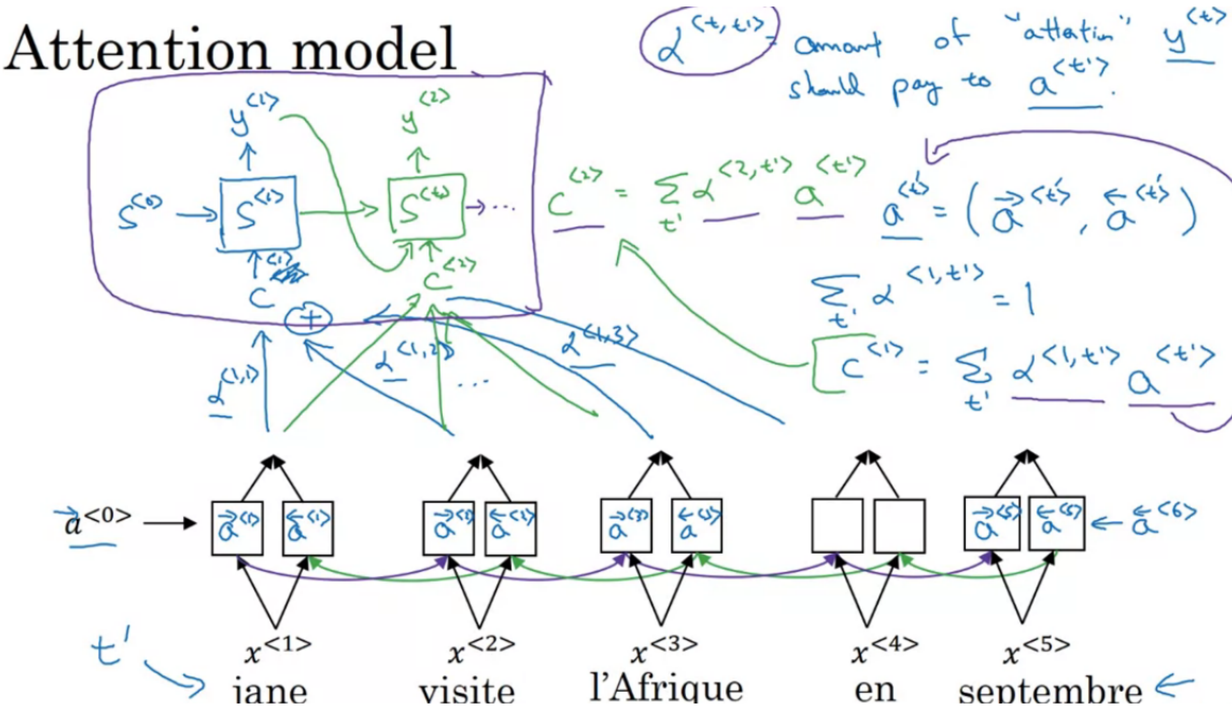
Attention (Alpha (t,x))→ How much weight to be used for generating t word using time-stamp x



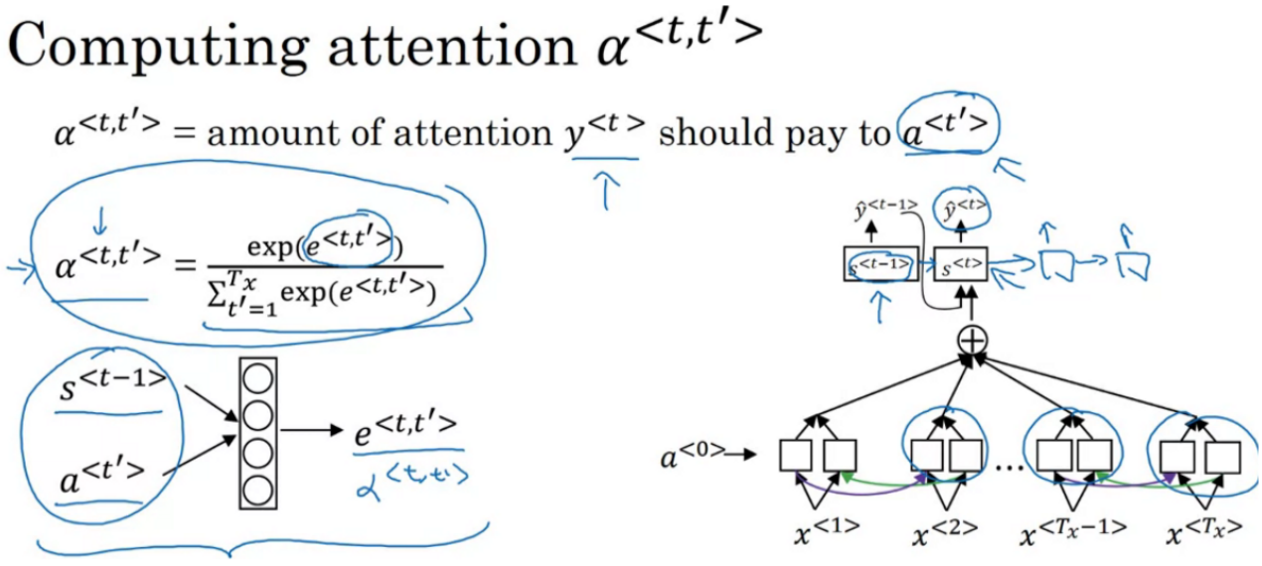
***Part — 01 (Attention)** → a is combination of backward and forward propagation .. For 1st word, will have 5 timestamp alphas i.e. attention weights and its summation will be 1. C (Context Vectors) is summation of different timestamps.*



PART — 02 (Attention) → A



Now, how to calculate Alpha (t,t') i.e. Amount of attention Y(t) should pay to a(t').

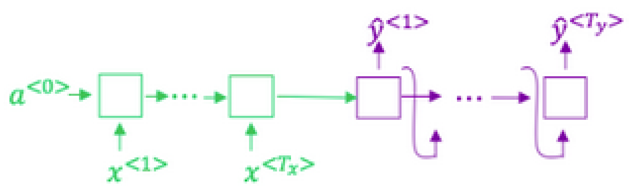


What is NEXT ? →https://workera.ai/?utm_source=coursera_sequence_models&utm_medium=Coursera&utm_campaign=coursera_sequence_models

https://drive.google.com/file/d/1099XMofOen_QfoNL3qqLUOXy-CdMyJQ4/view

QUIZ

1. Consider using this encoder-decoder model for machine translation.



This model is a "conditional language model" in the sense that the encoder portion (shown in green) is modeling the probability of the input sentence x.

- ○ True
- ◉ False

✓ Correct

2. In beam search, if you increase the beam width B, which of the following would you expect to be true? Check all that apply.

- [x] Beam search will run more slowly.

✓ **Correct**

- [x] Beam search will use up more memory.

✓ **Correct**

- [x] Beam search will generally find better solutions (i.e. do a better job maximizing $P(y \mid x)$)

✓ **Correct**

- [ ] Beam search will converge after fewer steps.

3. In machine translation, if we carry out beam search without using sentence normalization, the algorithm will tend to output overly short translations.

- (●) True
- ( ) False

✓ **Correct**

4. Suppose you are building a speech recognition system, which uses an RNN model to map from audio clip x to a text transcript y. Your algorithm uses beam search to try to find the value of y that maximizes $P(y \mid x)$.

   On a dev set example, given an input audio clip, your algorithm outputs the transcript $\hat{y} =$ "I'm building an A Eye system in Silly con Valley.", whereas a human gives a much superior transcript $y^* =$ "I'm building an AI system in Silicon Valley."

   According to your model,

   $P(\hat{y} \mid x) = 1.09 * 10^-7$

   $P(y^* \mid x) = 7.21 * 10^-8$

   Would you expect increasing the beam width B to help correct this example?

   - (●) No, because $P(y^* \mid x) \leq P(\hat{y} \mid x)$ indicates the error should be attributed to the RNN rather than to the search algorithm.
   - ( ) No, because $P(y^* \mid x) \leq P(\hat{y} \mid x)$ indicates the error should be attributed to the search algorithm rather than to the RNN.
   - ( ) Yes, because $P(y^* \mid x) \leq P(\hat{y} \mid x)$ indicates the error should be attributed to the RNN rather than to the search algorithm.
   - ( ) Yes, because $P(y^* \mid x) \leq P(\hat{y} \mid x)$ indicates the error should be attributed to the search algorithm rather than to the RNN.

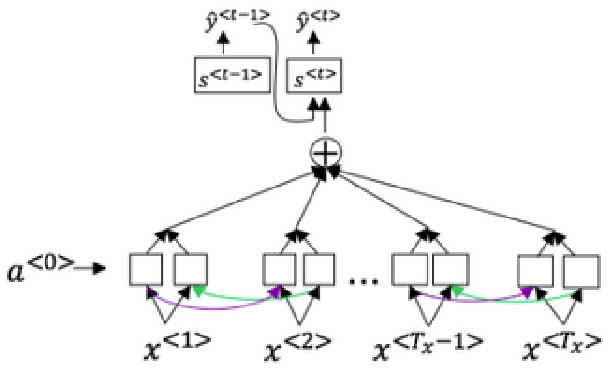
   ✓ **Correct**

5. Continuing the example from Q4, suppose you work on your algorithm for a few more weeks, and now find that for the vast majority of examples on which your algorithm makes a mistake, $P(y^* \mid x) > P(\hat{y} \mid x)$. This suggest you should focus your attention on improving the search algorithm.

   - (●) True.
   - ( ) False.

   ✓ **Correct**

6. Consider the attention model for machine translation.



Further, here is the formula for $\alpha^{<t,t'>}$.

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

Which of the following statements about $\alpha^{<t,t'>}$ are true? Check all that apply.

- [x] We expect $\alpha^{<t,t'>}$ to be generally larger for values of $a^{<t'>}$ that are highly relevant to the value the network should output for $y^{<t>}$. (Note the indices in the superscripts.)

Correct

- [ ] We expect $\alpha^{<t,t'>}$ to be generally larger for values of $a^{<t>}$ that are highly relevant to the value the network should output for $y^{<t'>}$. (Note the indices in the superscripts.)
- [ ] $\sum_t \alpha^{<t,t'>} = 1$ (Note the summation is over t.)
- [x] $\sum_{t'} \alpha^{<t,t'>} = 1$ (Note the summation is over t'.)

Correct

7. The network learns where to "pay attention" by learning the values $e^{<t,t'>}$, which are computed using a small neural network:

We can't replace $s^{<t-1>}$ with $s^{<t>}$ as an input to this neural network. This is because $s^{<t>}$ depends on $\alpha^{<t,t'>}$ which in turn depends on $e^{<t,t'>}$; so at the time we need to evalute this network, we haven't computed $s^{<t>}$ yet.

- (selected) True
- False

Correct

8. Compared to the encoder-decoder model shown in Question 1 of this quiz (which does not use an attention mechanism), we expect the attention model to have the greatest advantage when:

- (selected) The input sequence length T_x is large.
- The input sequence length T_x is small.

Correct

9. Under the CTC model, identical repeated characters not separated by the "blank" character (_) are collapsed. Under the CTC model, what does the following string collapse to?

__c_oo_o_kk___b_ooooo__oo__kkk

- cokbok
- (selected) cookbook
- cook book
- coookkboooooookkk

Correct

10. In trigger word detection, $x^{<t>}$ is:

- (selected) Features of the audio (such as spectrogram features) at time t.
- The t-th input word, represented as either a one-hot vector or a word embedding.
- Whether the trigger word is being said at time t.
- Whether someone has just finished saying the trigger word at time t.

Correct

Assignment → Jupyter Notebook

1 - Translating human readable dates into machine readable dates

- The model you will build here could be used to translate from one language to another, such as translating from English to Hindi.
- However, language translation requires massive datasets and usually takes days of training on GPUs.
- To give you a place to experiment with these models without using massive datasets, we will perform a simpler "date translation" task.
- The network will input a date written in a variety of possible formats (*e.g. "the 29th of August 1958", "03/30/1968", "24 JUNE 1987"*)
- The network will translate them into standardized, machine readable dates (*e.g. "1958-08-29", "1968-03-30", "1987-06-24"*).
- We will have the network learn to output dates in the common machine-readable format YYYY-MM-DD.

1.1 - Dataset

We will train the model on a dataset of 10,000 human readable dates and their equivalent, standardized, machine readable dates. Let's run the following cells to load the dataset and print some examples.

```
dataset[:10]
```

```
[('9 may 1998', '1998-05-09'),
 ('10.11.19', '2019-11-10'),
 ('9/10/70', '1970-09-10'),
 ('saturday april 28 1990', '1990-04-28'),
 ('thursday january 26 1995', '1995-01-26'),
 ('monday march 7 1983', '1983-03-07'),
 ('sunday may 22 1988', '1988-05-22'),
 ('08 jul 2008', '2008-07-08'),
 ('8 sep 1999', '1999-09-08'),
 ('thursday january 1 1981', '1981-01-01')]
```

You've loaded:

- `dataset`: a list of tuples of (human readable date, machine readable date).
- `human_vocab`: a python dictionary mapping all characters used in the human readable dates to an integer-valued index.
- `machine_vocab`: a python dictionary mapping all characters used in machine readable dates to an integer-valued index.
    - **Note**: These indices are not necessarily consistent with `human_vocab`.
- `inv_machine_vocab`: the inverse dictionary of `machine_vocab`, mapping from indices back to characters.

Let's preprocess the data and map the raw text data into the index values.

- We will set Tx=30
    - We assume Tx is the maximum length of the human readable date.
    - If we get a longer input, we would have to truncate it.
- We will set Ty=10
    - "YYYY-MM-DD" is 10 characters long.

You now have:

- X: a processed version of the human readable dates in the training set.
    - Each character in X is replaced by an index (integer) mapped to the character using `human_vocab`.
    - Each date is padded to ensure a length of T_x using a special character (< pad >).
    - `X.shape = (m, Tx)` where m is the number of training examples in a batch.
- Y: a processed version of the machine readable dates in the training set.
    - Each character is replaced by the index (integer) it is mapped to in `machine_vocab`.
    - `Y.shape = (m, Ty)`.
- Xoh: one-hot version of X
    - Each index in X is converted to the one-hot representation (if the index is 2, the one-hot version has the index position 2 set to 1, and the remaining positions are 0.
    - `Xoh.shape = (m, Tx, len(human_vocab))`
- Yoh: one-hot version of Y
    - Each index in Y is converted to the one-hot representation.
    - `Yoh.shape = (m, Tx, len(machine_vocab))`.
    - `len(machine_vocab) = 11` since there are 10 numeric digits (0 to 9) and the `-` symbol.

- Let's also look at some examples of preprocessed training examples.
- Feel free to play with `index` in the cell below to navigate the dataset and see how source/target dates are preprocessed.

```python
index = 0
print("Source date:", dataset[index][0])
print("Target date:", dataset[index][1])
print()
print("Source after preprocessing (indices):", X[index])
print("Target after preprocessing (indices):", Y[index])
print()
print("Source after preprocessing (one-hot):", Xoh[index])
print("Target after preprocessing (one-hot):", Yoh[index])
```

```
Source date: 9 may 1998
Target date: 1998-05-09

Source after preprocessing (indices): [12  0 24 13 34  0  4 12 12 11 36 36 36 36 36 36 36 36 36 36 36 36 36 3
6
 36 36 36 36 36]
Target after preprocessing (indices): [ 2 10 10  9  0  1  6  0  1 10]

Source after preprocessing (one-hot): [[ 0.  0.  0. ...,  0.  0.  0.]
 [ 1.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  0.]
 ...,
 [ 0.  0.  0. ...,  0.  0.  1.]
 [ 0.  0.  0. ...,  0.  0.  1.]
 [ 0.  0.  0. ...,  0.  0.  1.]]
Target after preprocessing (one-hot): [[ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
 [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]]
```

2.1 - Attention mechanism

In this part, you will implement the attention mechanism presented in the lecture videos.

- Here is a figure to remind you how the model works.
  - The diagram on the left shows the attention model.
  - The diagram on the right shows what one "attention" step does to calculate the attention variables $\alpha^{\langle t, t' \rangle}$.
  - The attention variables $\alpha^{\langle t, t' \rangle}$ are used to compute the context variable $context^{\langle t \rangle}$ for each timestep in the output ($t = 1, \dots, T_y$).

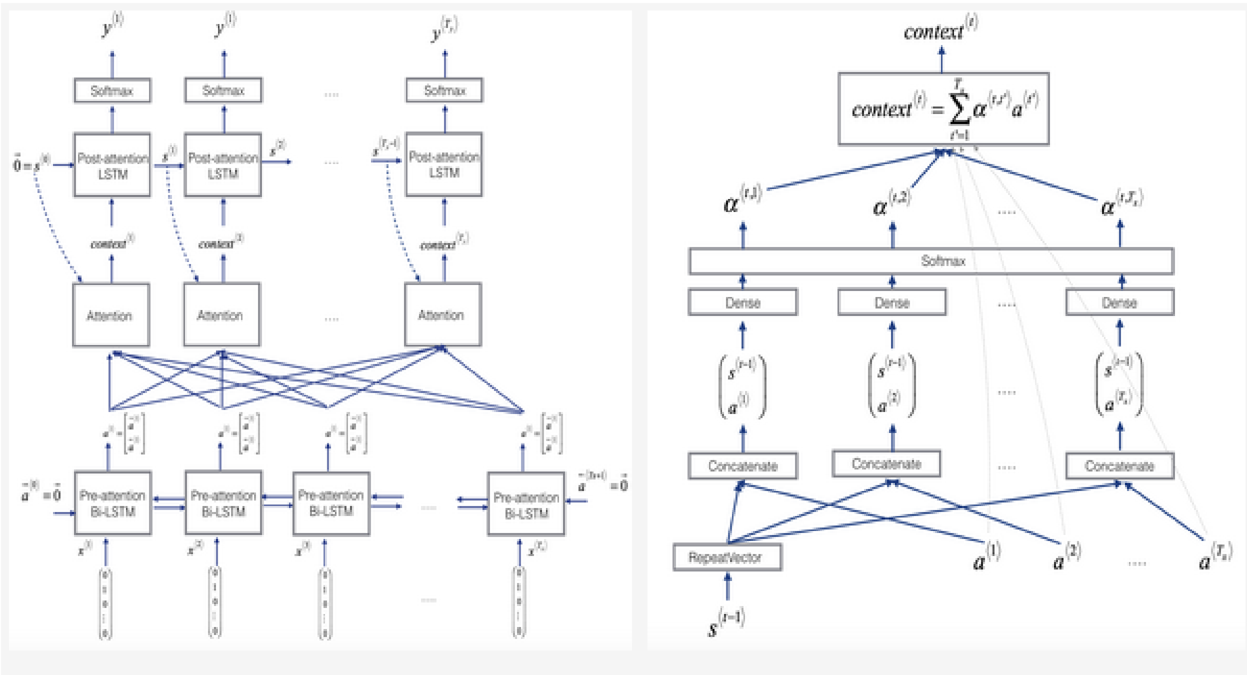


Figure 1: Neural machine translation with attention