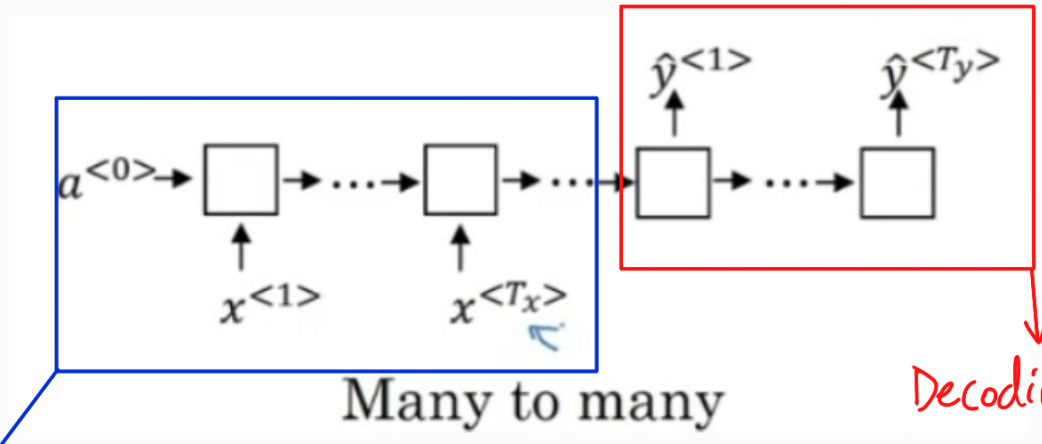
The sequence to sequence model, is a many to many RNN.



Decoding network

Encoding network

So, we are computing $P(y^{<1>}, \ldots, y^{<T_y>} | x^{<1>}, \ldots, x^{<Tx>})$,

($y^{<1>}, \ldots, y^{<T_y>}$: output sentence; $x^{<1>}, \ldots, x^{<Tx>}$: Input sentence)

and try to get the $\underset{y^{<1>}, \ldots, y^{<T_y>}}{\text{argmax}} P(y^{<1>}, \ldots, y^{<T_y>} | x)$.

We don't do a random search, but a beam search.

- Beam search: we specify a parameter B, which represents the number of most probable words to select from $\hat{y}^{<i>}$ and pass to $\hat{y}^{<i+1>}$.

For instance, after the encoding layer computations have finished, we get the first $\hat{y}$ from softmax. Select the B max, and pass them to the next.

Consequently, at each step we instantiate B copies of the network.

It can be optimized:

- Length normalization: avoid really short translations

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

⇓

prone to numerical underflow!!

Take logs:

$$\arg\max_{y} \sum_{y=1}^{T_y} \log P(y^{<t>} | x, y^{<1>}, \ldots, y^{<t-1>})$$

As the logarithmic function is a strictly monotonically increasing function, we know that maximizing $\log(P(y|x))$ = maximizing $P(y|x)$

We can then normalize (divide by T_y^{α}):

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

The α is for softening.

- Choosing a great B:
  - ↳ Large B ⇒ better results, slower
  - ↳ Small B ⇒ worse results, faster

Note ⇒ Bim search does NOT always guarantee obtaining the best result.

Error analysis on beam search

$P(y^*|x)$

$P(\hat{y}|x)$

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

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

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

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.

(Being y^* the output of a human).

A sentence may be translated in different equally correct ways. How to choose?

BLEU score: how similar it is to human translation.

For n-grams;

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

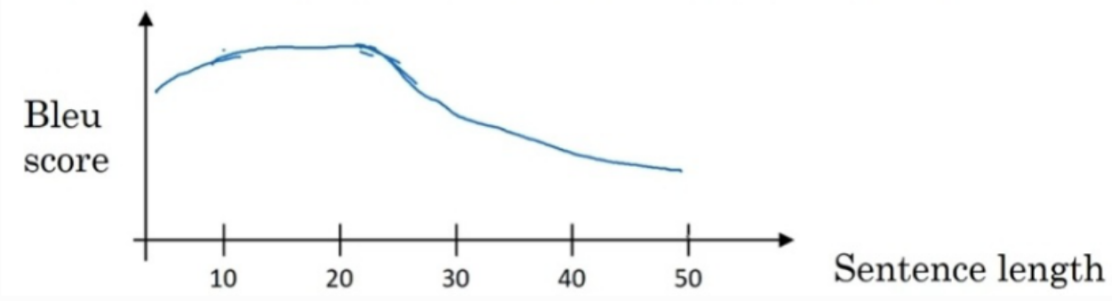
→ max times that the n-gram appears for all the human-translated sample

↳ how many times a n-gram appears

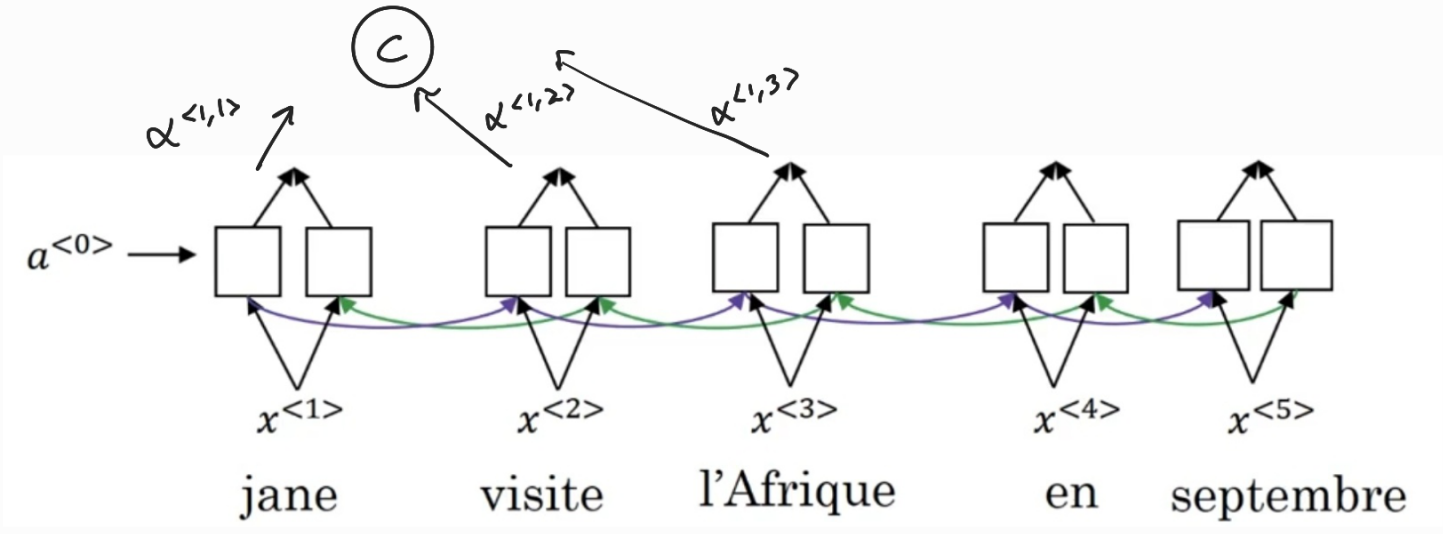
Combining Blens: $BP \cdot e^{(\text{mean of blens})}$

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

the encoder-decoder model decreases performance as the length of the sentence to translate is bigger.



Attention models: with a BRNN, we compute some α weights for each node, and then compute a context var (c).



this way, we are just paying attention at a part of the sentence.

Each alpha expresses how much of the context will depend on the activations of each time step $(\overleftarrow{a}^{<t>}, \overrightarrow{a}^{<t>})$. therefore, it is the attention $y^{<t>}$ should pay to $a^{<t'>}$.

$$c^{<i>} = \sum_{t'} \alpha^{<i,t'>} a^{<t'>} \quad ; \text{ being } a^{<t'>} = (\overrightarrow{a}^{<t'>}, \overleftarrow{a}^{<t'>})$$

But, how do we compute each α?

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

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

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