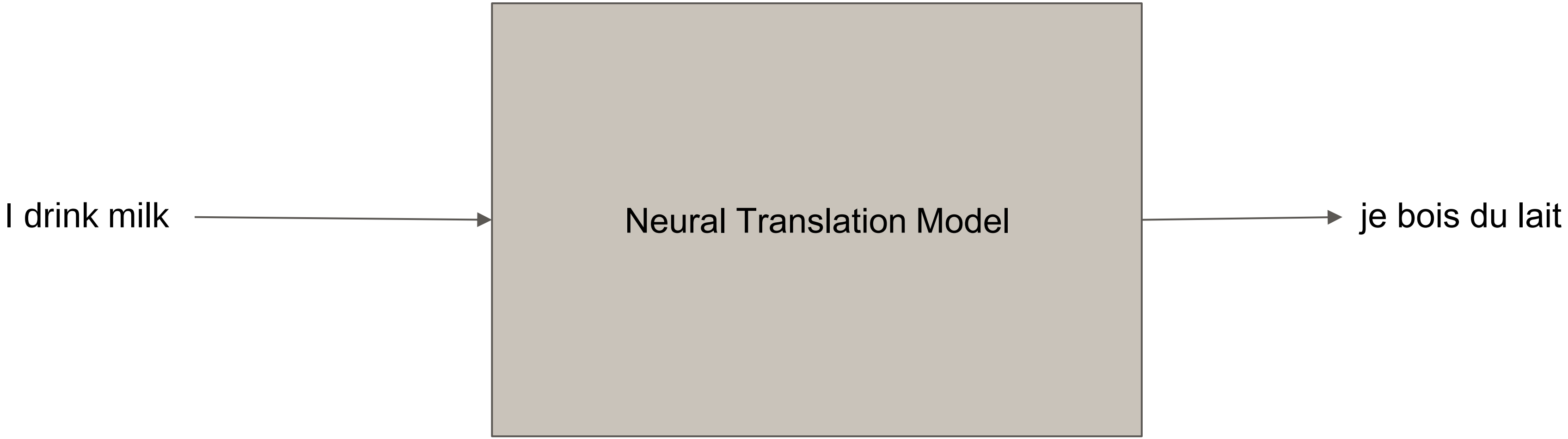
An Encoder–Decoder Network for Neural Machine Translation

Neural Machine Translation

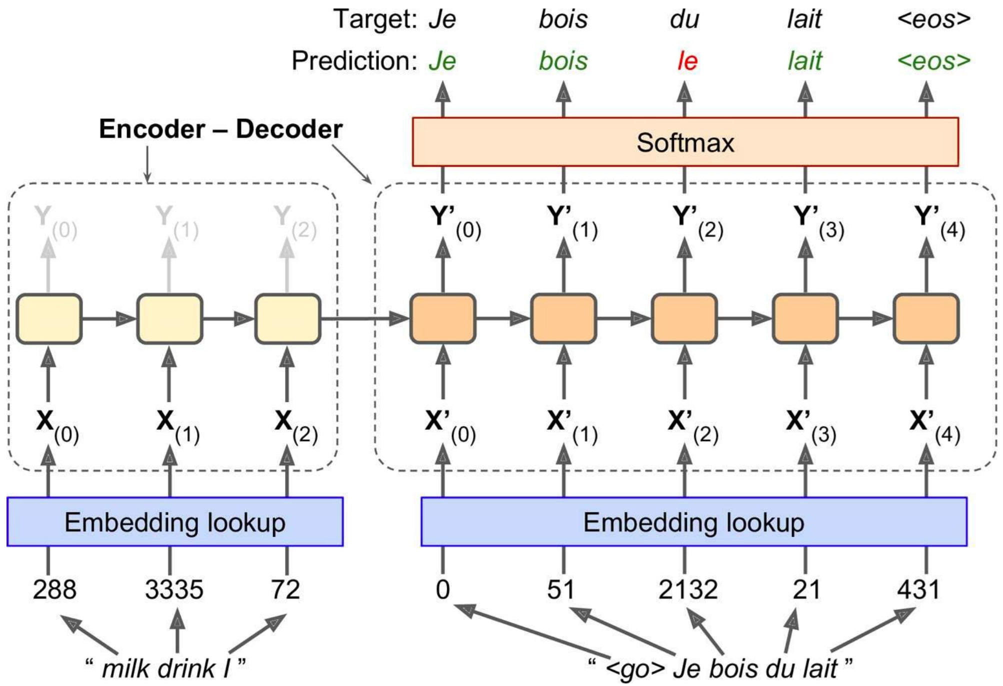
A simple Machine Translation Model that translates English sentences to French



How will you build it?

Encoder–Decoder Network for Neural Machine Translation

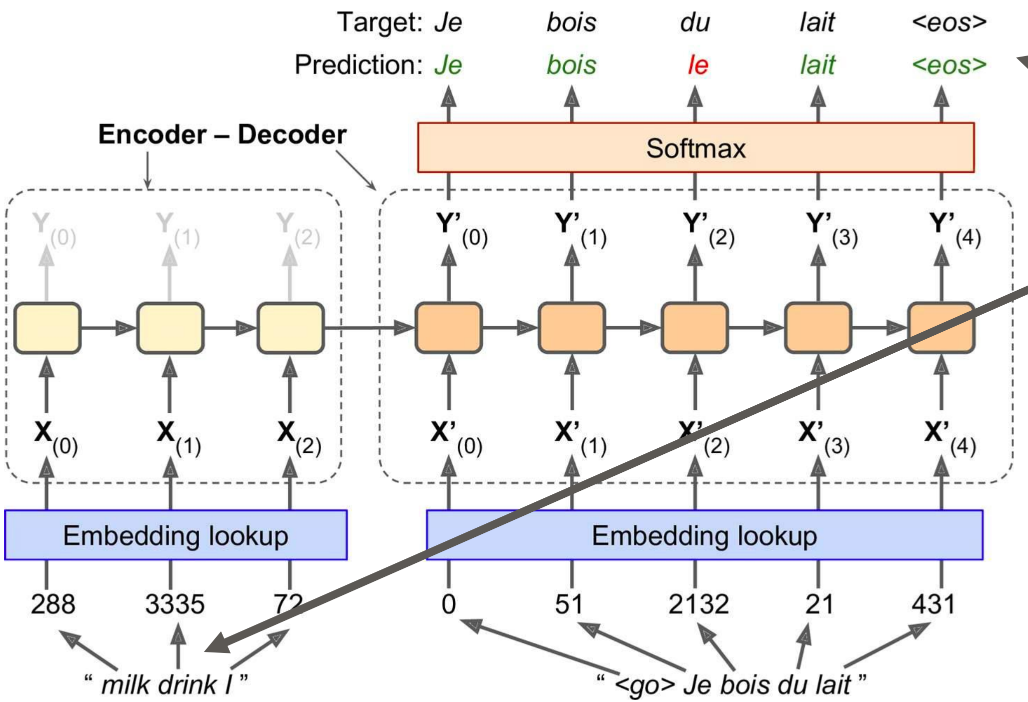
A simple Machine Translation Model that translates English sentences to French



Encoder–Decoder Network for Neural Machine Translation

Let's learn how this Encoder–Decoder Network
for Machine Translation is **trained**

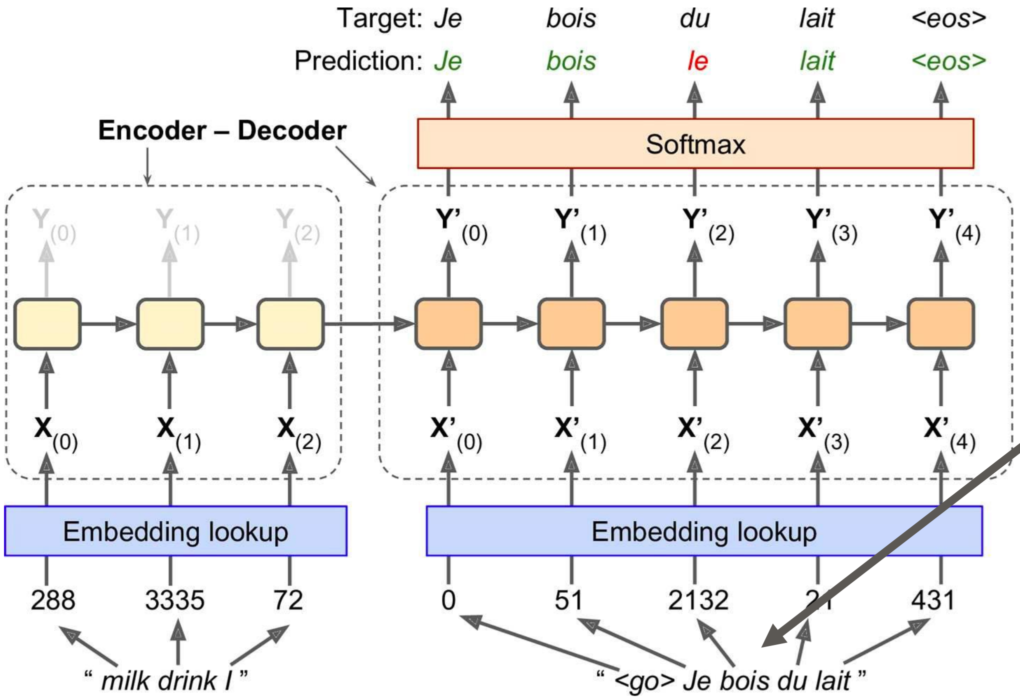
Encoder–Decoder Network for Neural Machine Translation



The English sentences are fed to the **encoder**

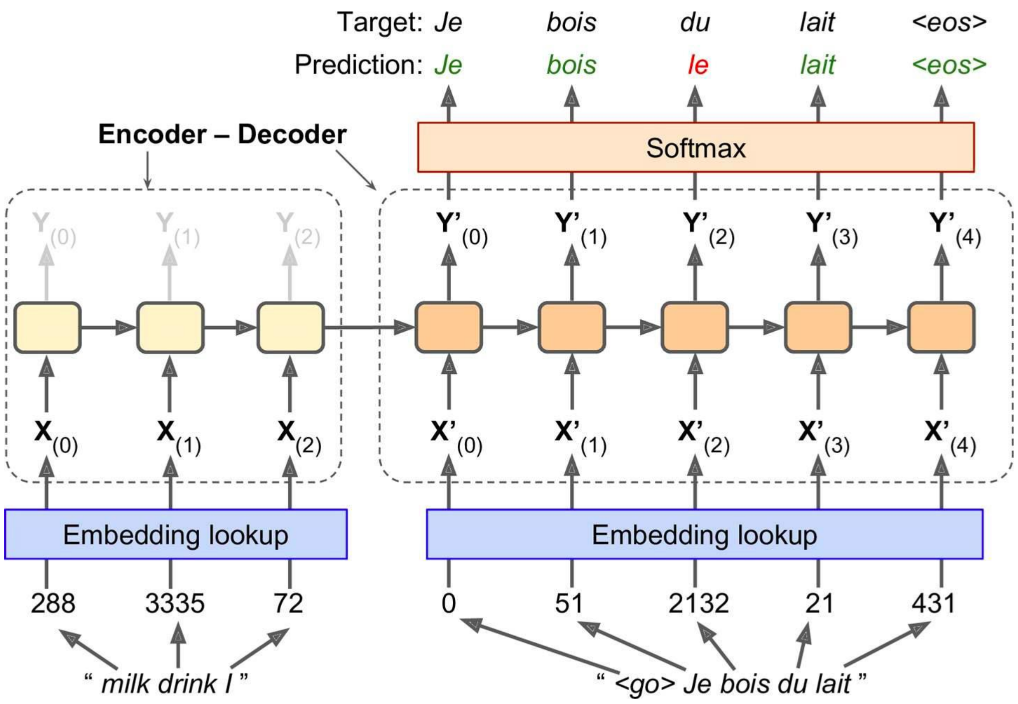
The **decoder** outputs the French translations

Encoder–Decoder Network for Neural Machine Translation



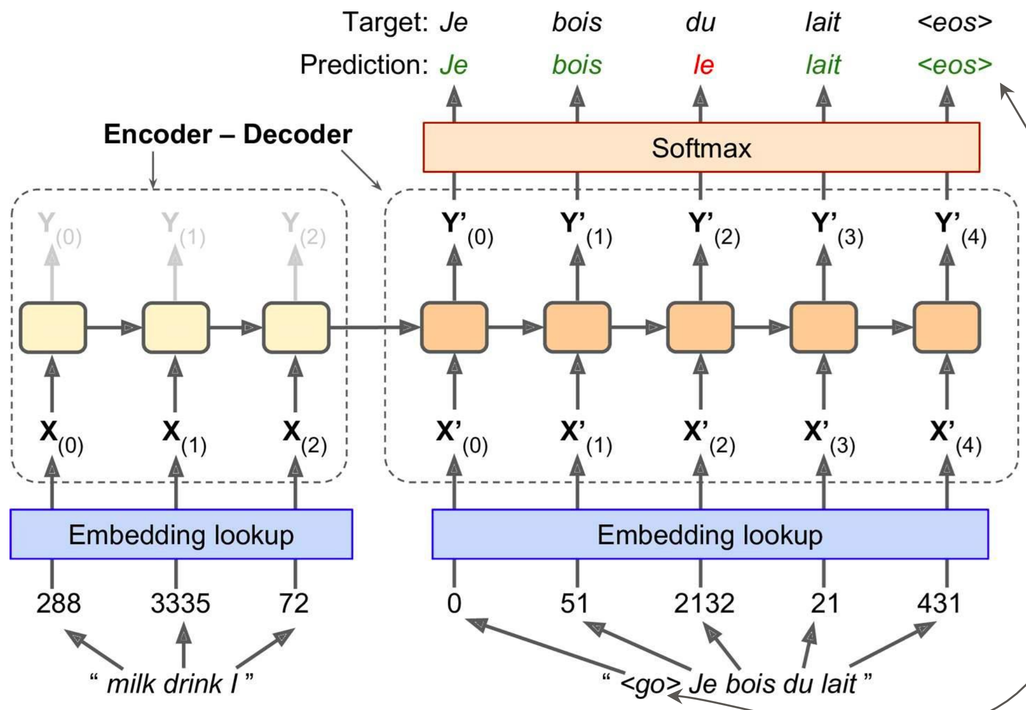
French translations are also used as inputs to the decoder pushed back by one step

Encoder–Decoder Network for Neural Machine Translation



Decoder is given as input the word that **it should** have output **at the previous step** regardless of what it actually output at the current step.

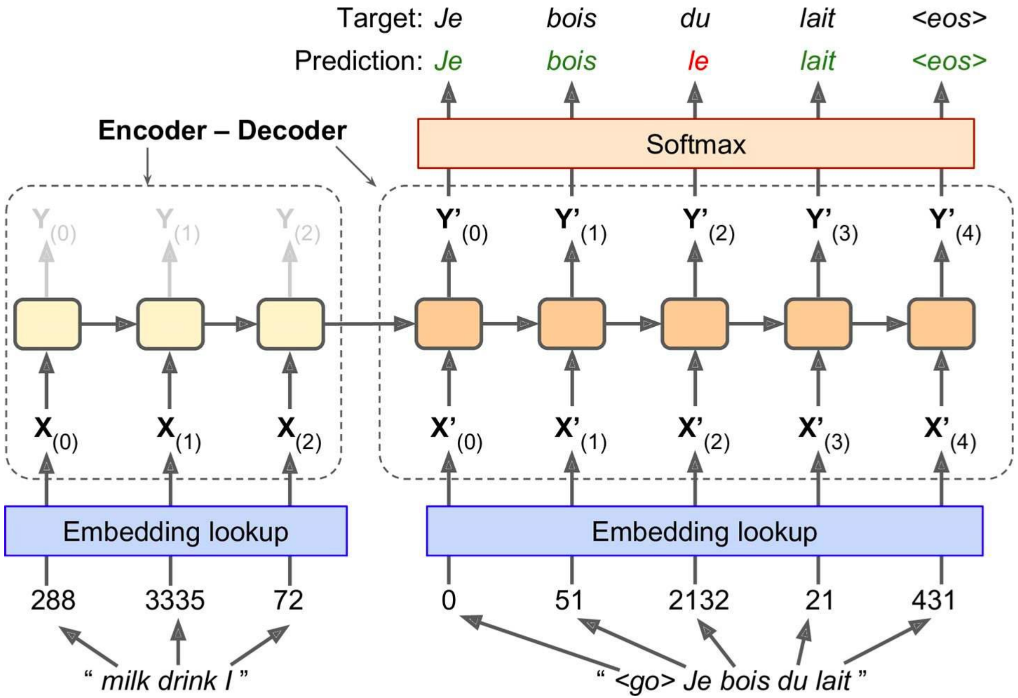
Encoder–Decoder Network for Neural Machine Translation



For the first word, **the decoder** is given token that represents the beginning of sentence (here, **"<go>"**)

The decoder is expected to end sentence with end-of-sequence (EOS) token (here, **"<eos>"**)

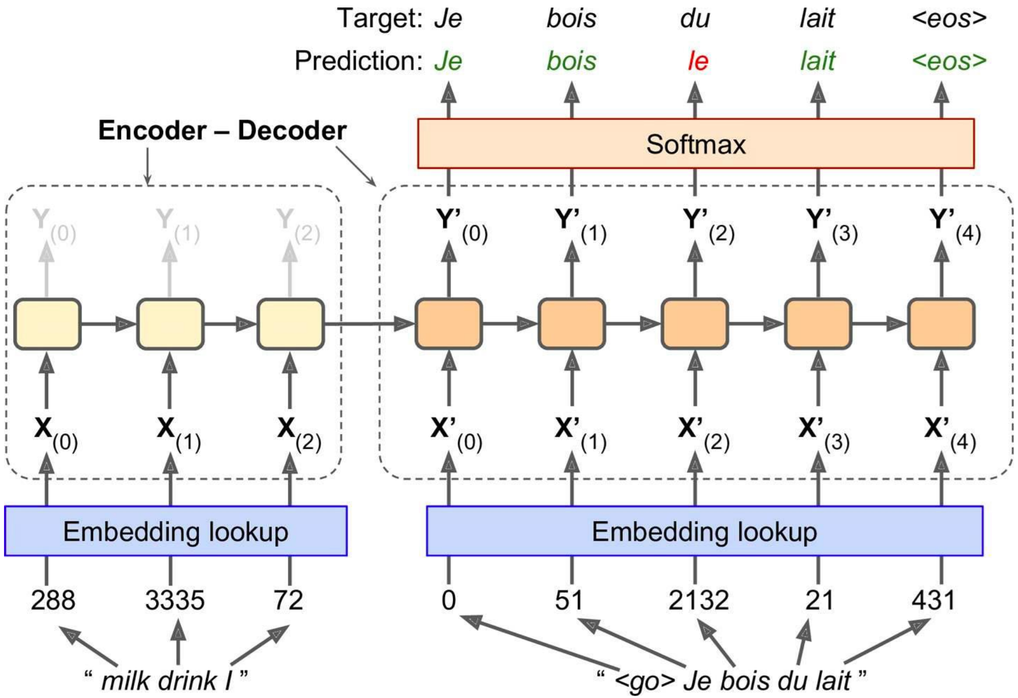
Encoder–Decoder Network for Neural Machine Translation



Question:
Why are the English sentences reversed before feeding it to the encoder??

Here **"I drink milk"** is reversed to **"milk drink I"**

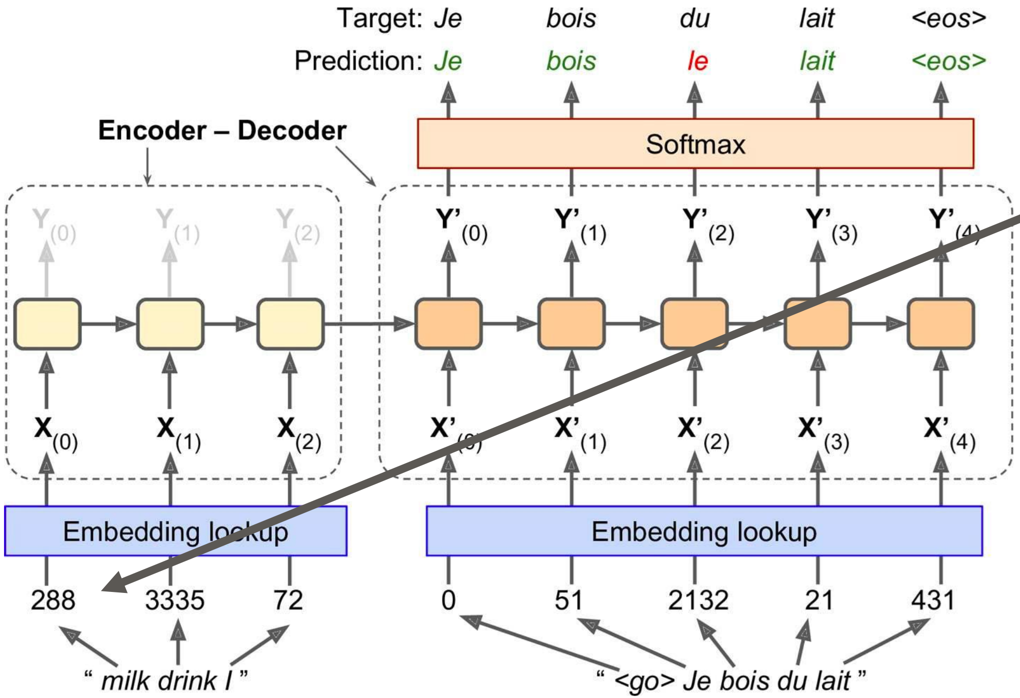
Encoder–Decoder Network for Neural Machine Translation



Answer:
So that the beginning of the English sentence will be fed last to the encoder.

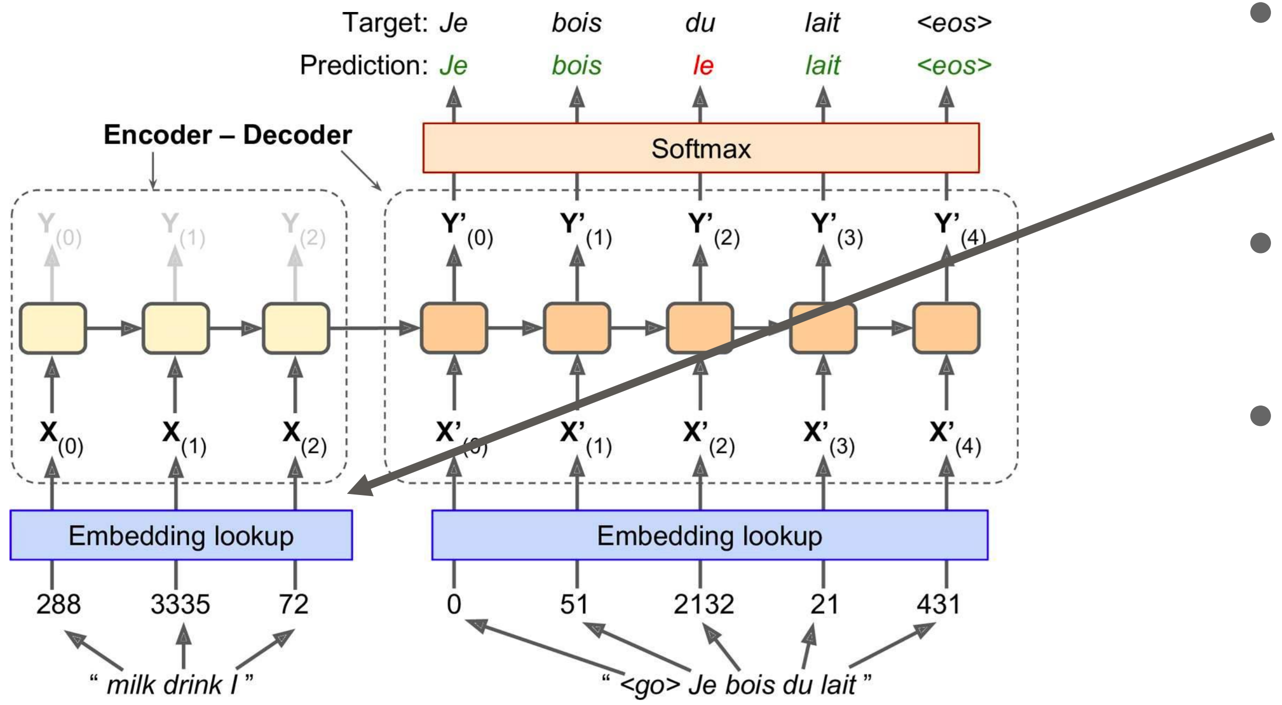
That's generally the first thing that the decoder needs to translate.

Encoder–Decoder Network for Neural Machine Translation



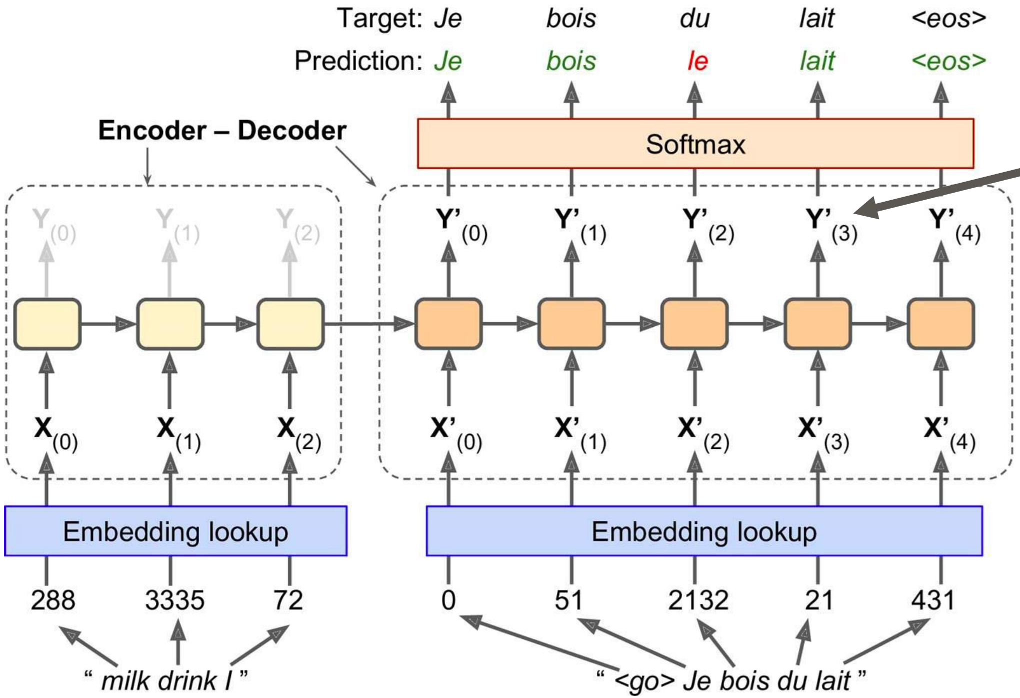
- Each word is initially represented by a simple integer identifier
- e.g., 288 for the word "milk"

Encoder–Decoder Network for Neural Machine Translation



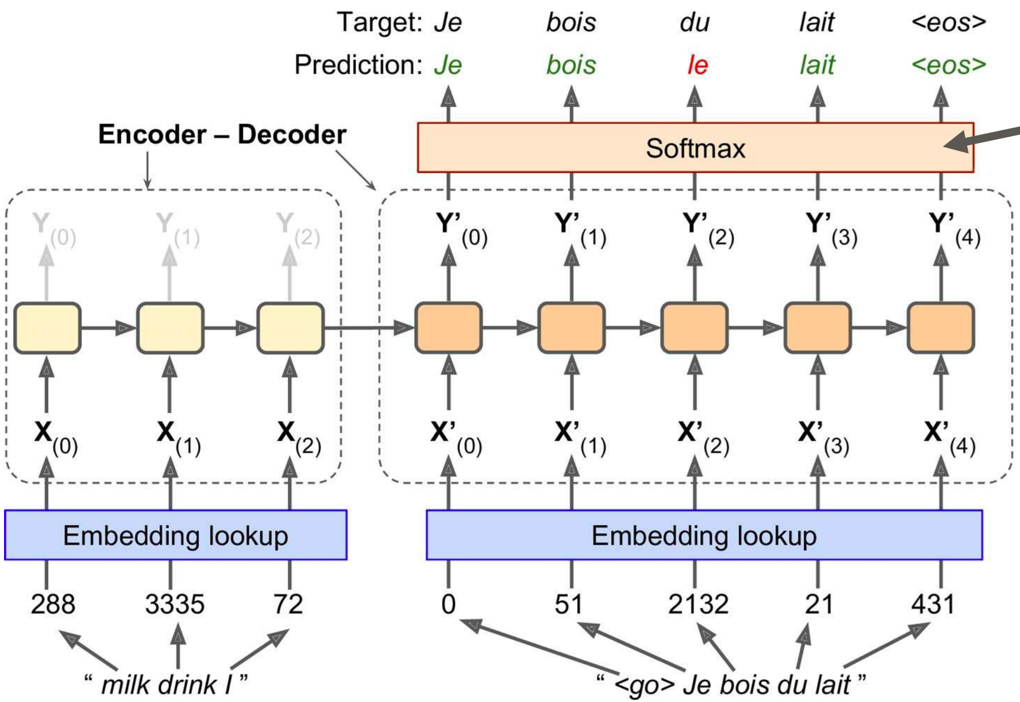
- Next, an embedding lookup returns the word embedding
- This is a dense, fairly low-dimensional vector
- These word embeddings are what is actually fed to the encoder and the decoder

Encoder–Decoder Network for Neural Machine Translation



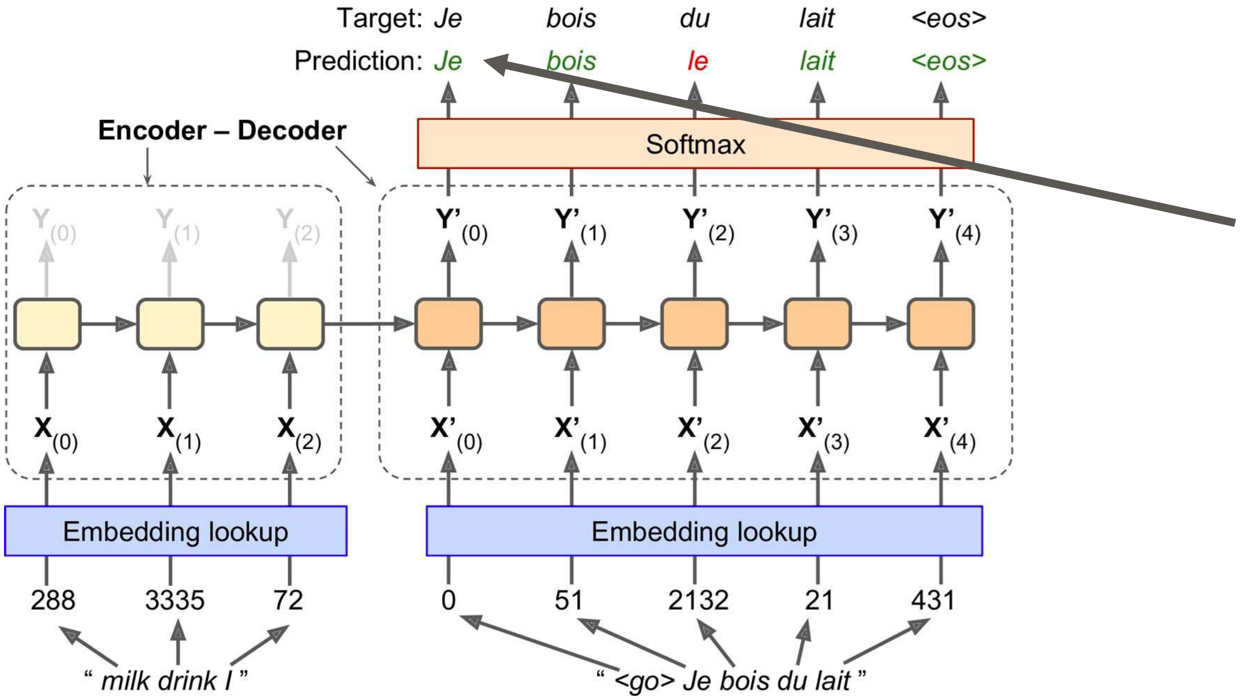
- At each step, the decoder outputs a score for each word in the output vocabulary i.e., French

Encoder–Decoder Network for Neural Machine Translation



- Then **Softmax layer** turns these scores into probabilities

Encoder–Decoder Network for Neural Machine Translation



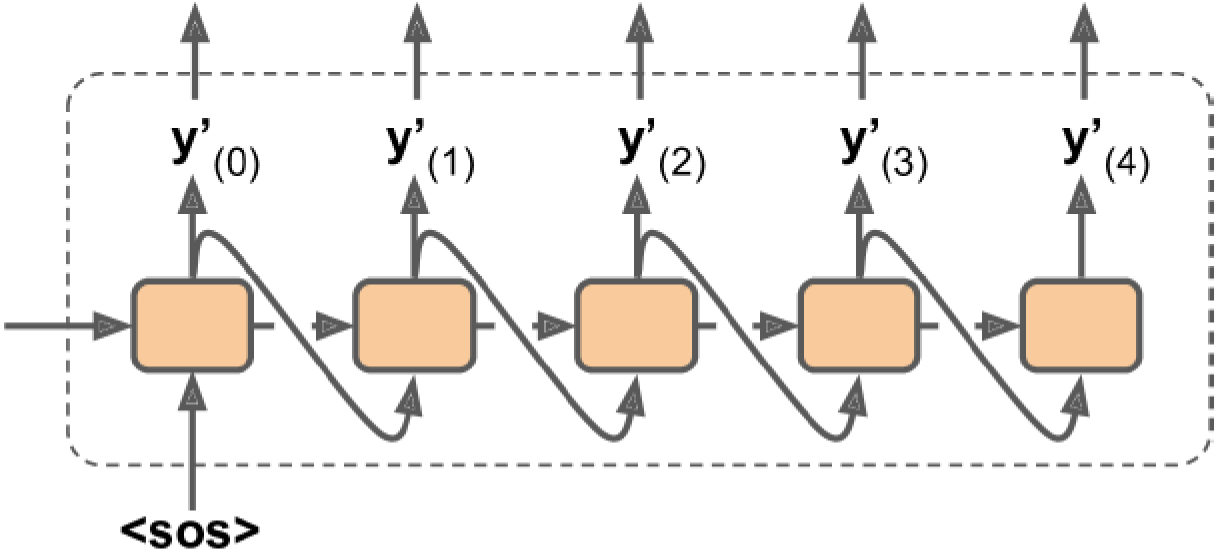
- For example, at the first step "Je" may have a probability of 20%, "Tu" may have a probability of 1%, and so on
- The word with the highest probability is output

Encoder–Decoder Network for Neural Machine Translation

How can we use this Encoder–Decoder Network for Machine Translation at the inference time, since we will not have the target sentence to feed to the decoder?

Encoder–Decoder Network for Neural Machine Translation

- Feed the decoder the word that it output at the previous step
- This requires an embedding lookup that is not shown on the diagram



But NMT with Encoder-Decoder Architecture was not so successful.

Encoder–Decoder Network for NMT Limitations

- Lack of Context Understanding
- Lengthy Sequences Challenge
- Global Context Overlooking

Attention is all you need!

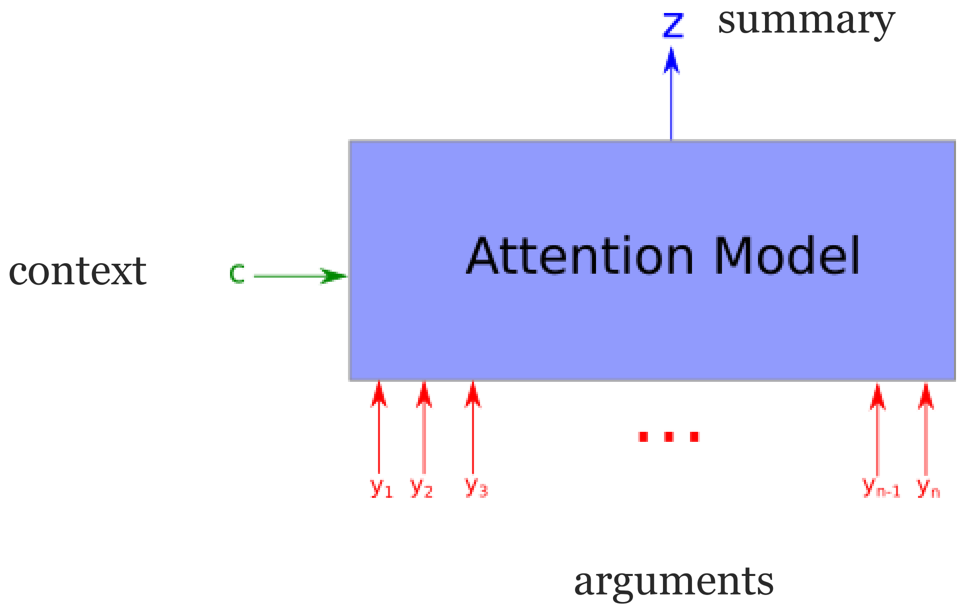
Attention Mechanisms

- In a 2014 paper, Dzmitry Bahdanau et al. introduced a technique that allowed the decoder to focus on the appropriate words (as encoded by the encoder) at each time step

Attention Mechanisms

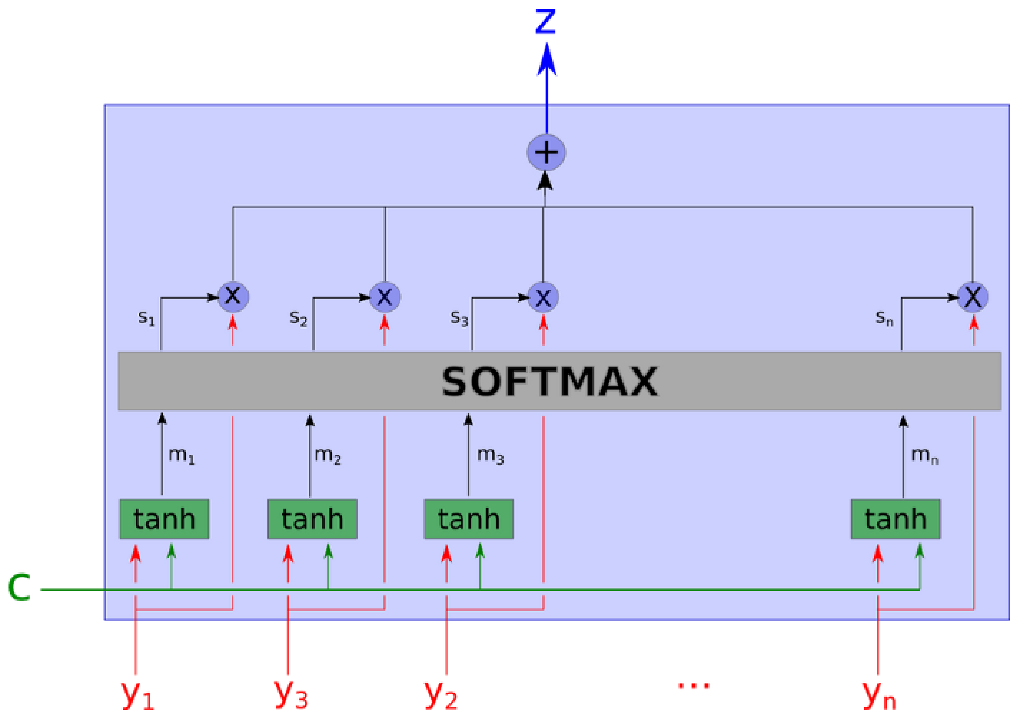
- So the path from an input word to its translation
    - Is now much shorter
    - So short-term memory limitations of RNNs have much less impact

Attention Mechanisms



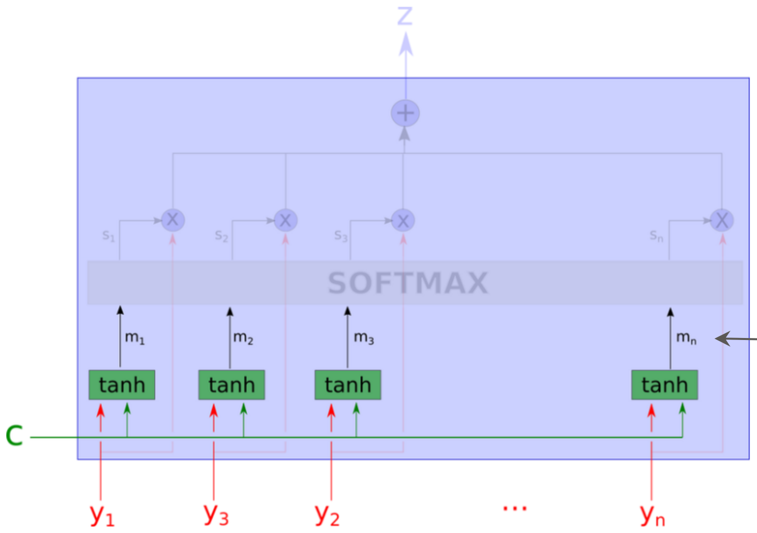
Source: https://medium.com/heuritech/attention-mechanism-5aba9a2d4727

Attention Mechanisms



Source: https://medium.com/heuritech/attention-mechanism-5aba9a2d4727

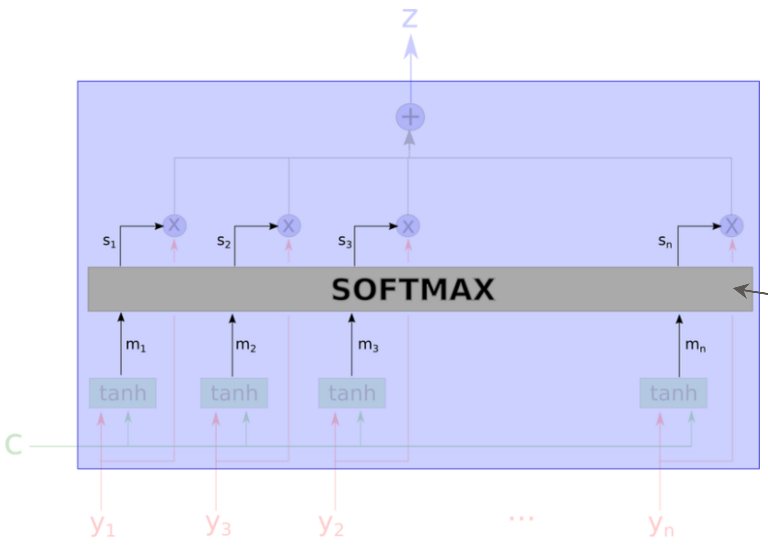
Attention Mechanisms



$m_i = \tanh(W_{cm}c + W_{ym}y_i)$

Source: https://medium.com/heuritech/attention-mechanism-5aba9a2d4727

Attention Mechanisms



When 1 output saying which input is max, like below –

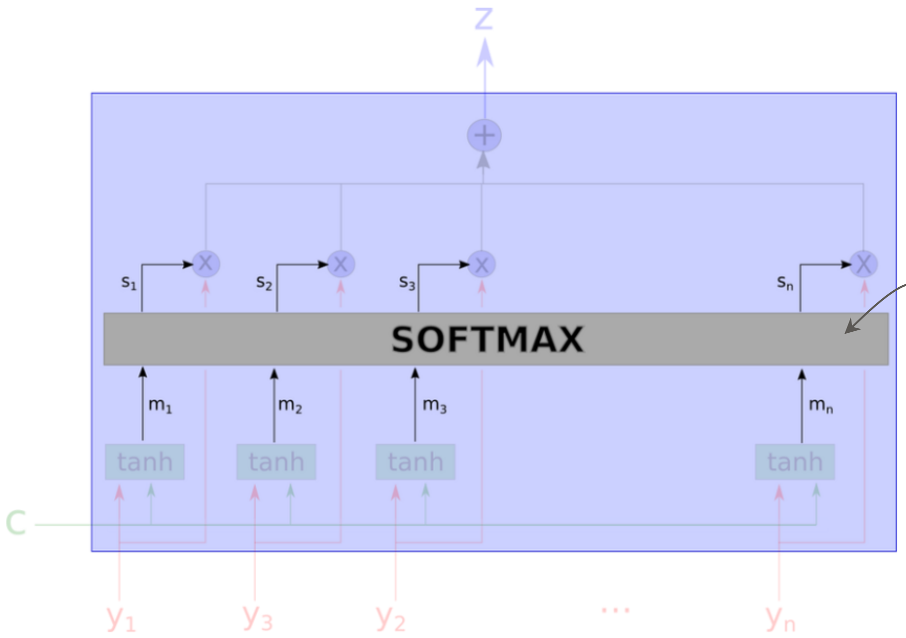
$\text{argmax}(x_1,....,x_n) = (0,....,0,1,0,....0)$

Then –

$\text{softmax}(x_1,....,x_n) = (e^{xi}/\sum e^{xj})_i$

Source: https://medium.com/heuritech/attention-mechanism-5aba9a2d4727

Attention Mechanisms



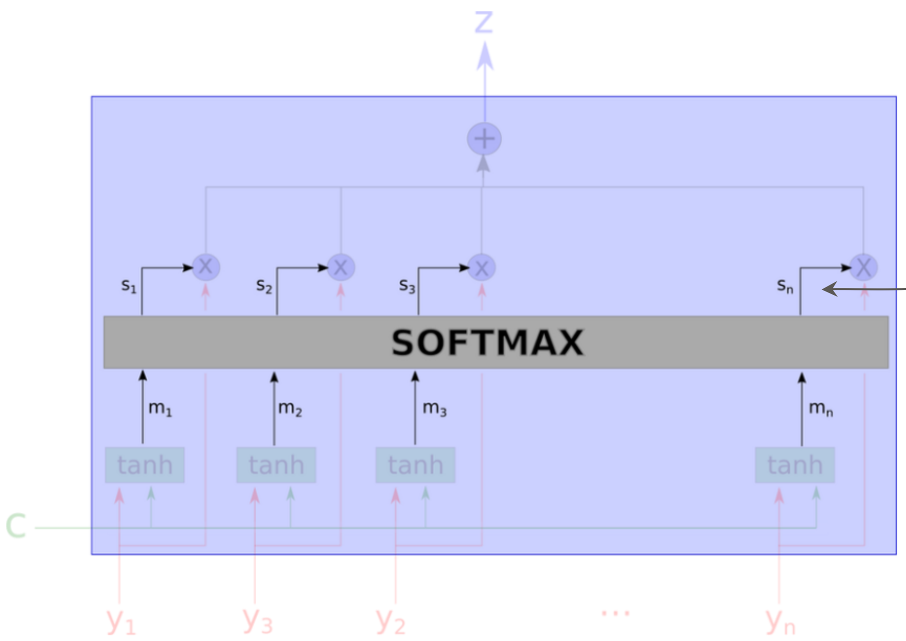
When one x_i is bigger than other –

Then –

$\text{softmax}(x_1,\ldots,x_n) \cong \text{argmax}(x_1,\ldots,x_n)$

Source: https://medium.com/heuritech/attention-mechanism-5aba9a2d4727

Attention Mechanisms



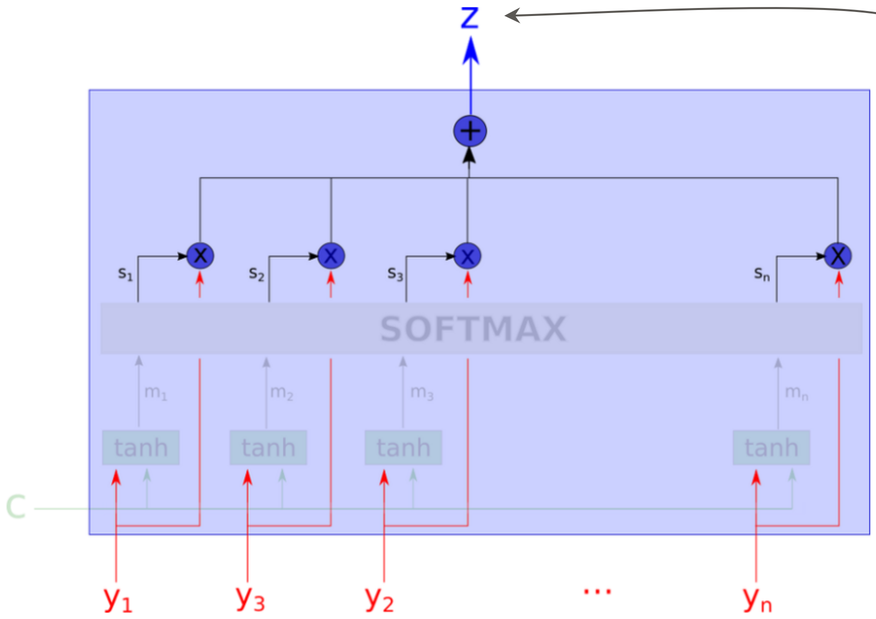
$s_i \propto \exp(\langle w_m, m_i \rangle)$

$\sum_i s_i = 1$

s_i are the softmax of the m_i projected on a learned direction

Source: https://medium.com/heuritech/attention-mechanism-5aba9a2d4727

Attention Mechanisms



$z = \sum_i s_i y_i$

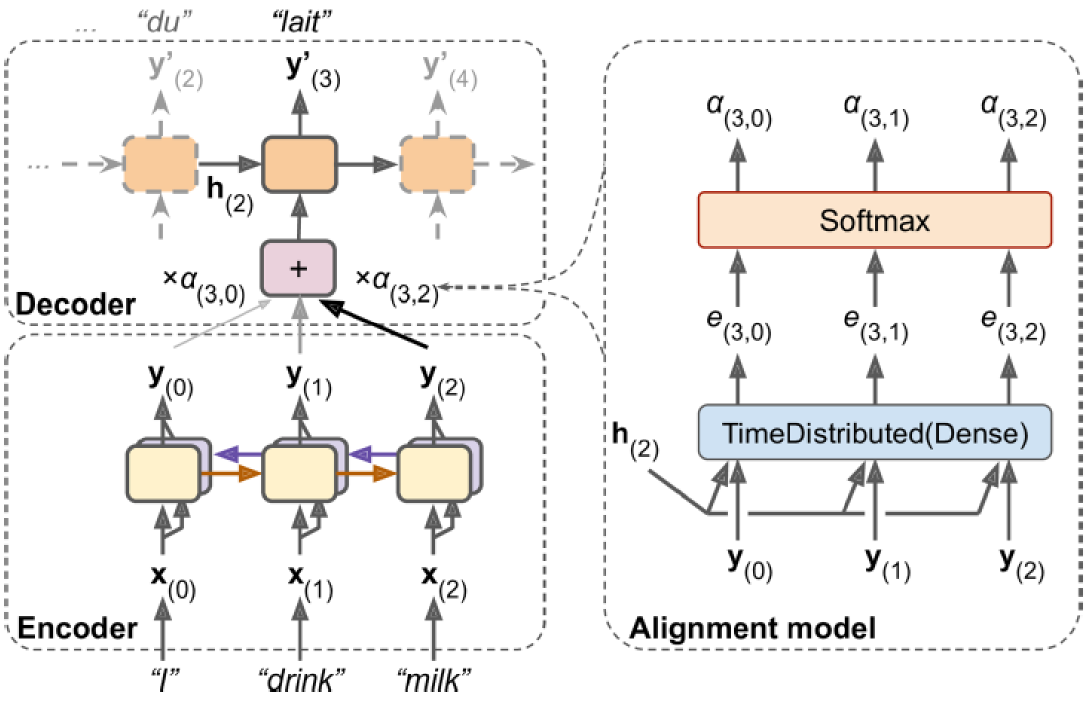
z is the weighted arithmetic mean of all the y_i

Source: https://medium.com/heuritech/attention-mechanism-5aba9a2d4727

Attention Mechanisms - References

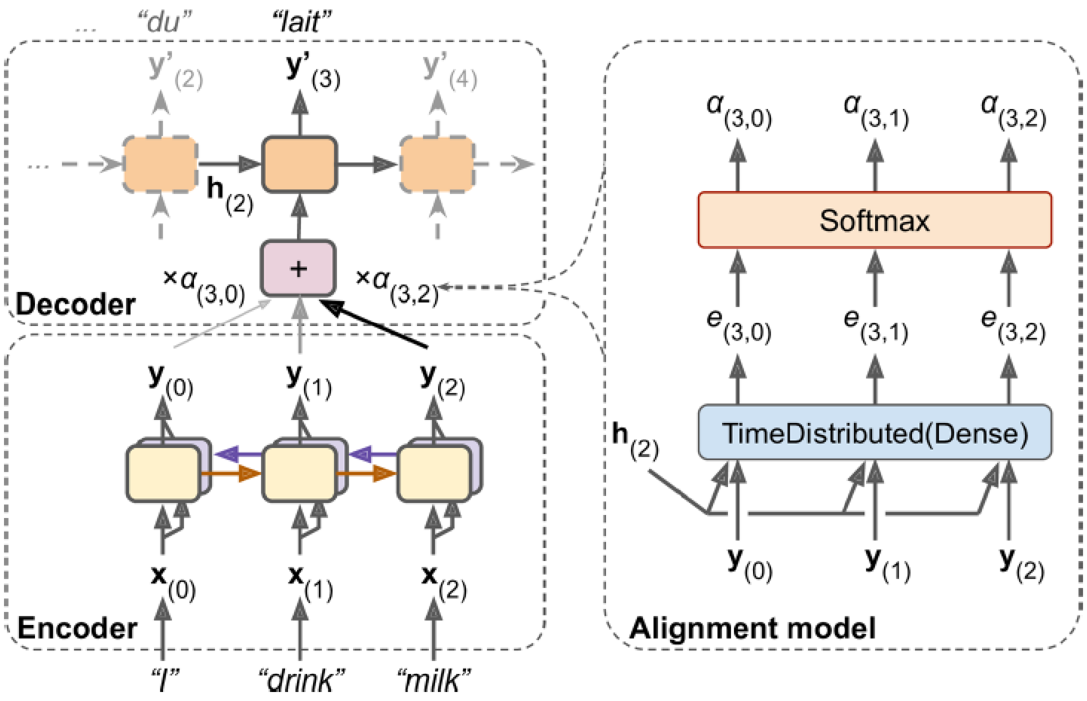
- https://arxiv.org/abs/1604.03736
- https://distill.pub/2016/augmented-rnns/
- http://akosiorek.github.io/ml/2017/10/14/visual-attention.html
- https://medium.com/heuritech/attention-mechanism-5aba9a2d4727

Attention Mechanisms



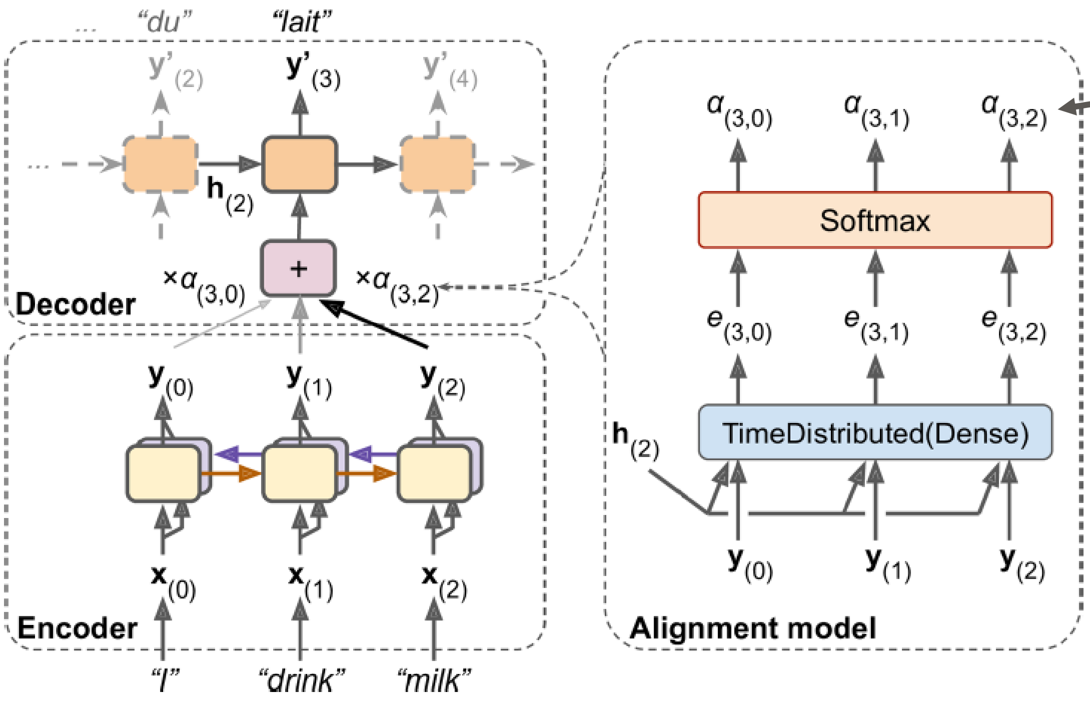
This shows this model's slightly simplified architecture

Attention Mechanisms



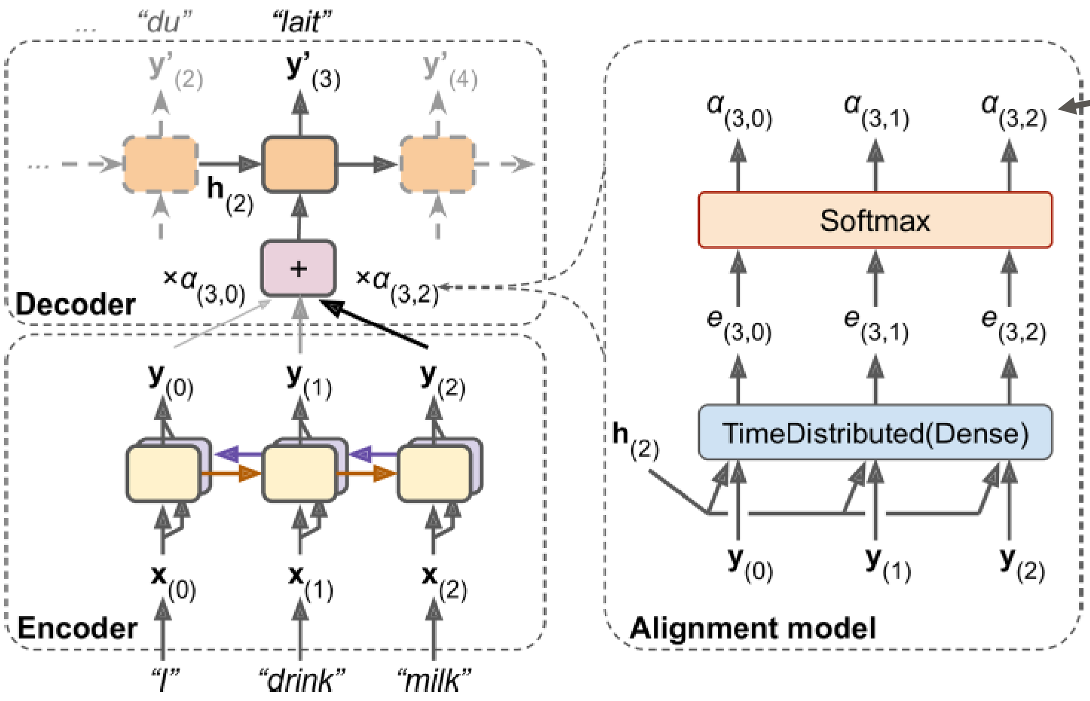
Instead of just sending the encoder's final hidden state to the decoder, we now send all of its outputs to the decoder

Attention Mechanisms



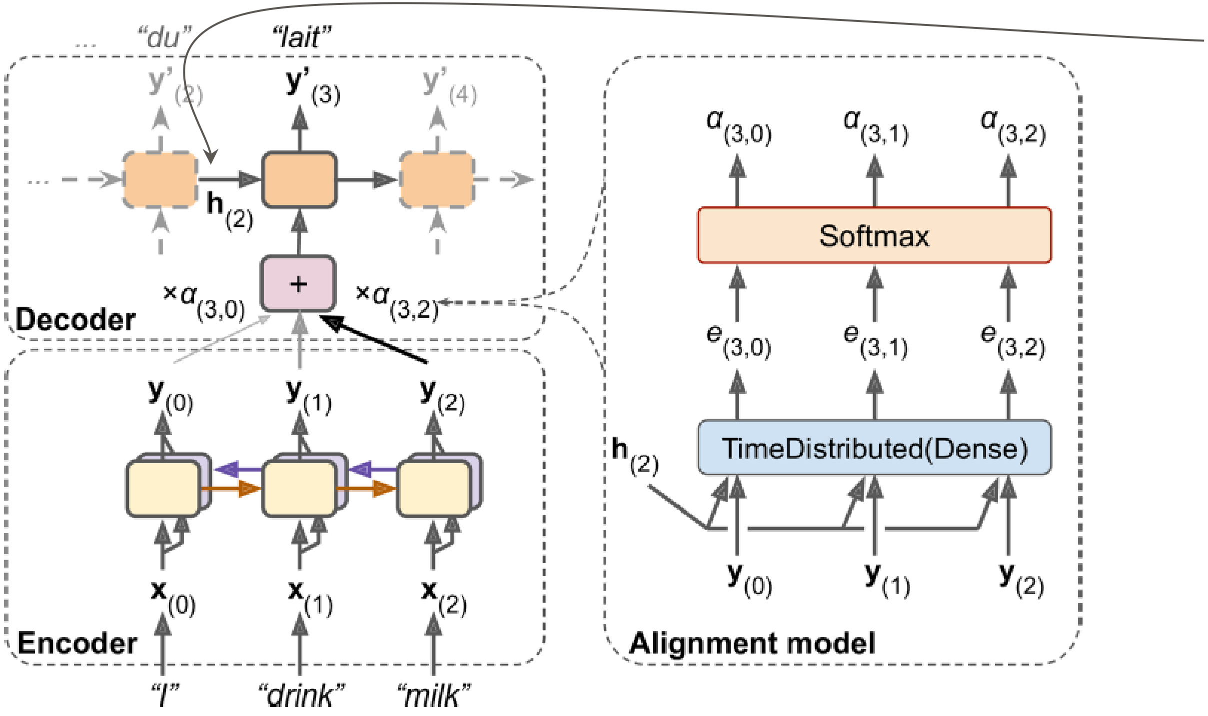
If the weight α(3,2) is much larger than the weights α(3,0) and α(3,1), then the decoder will pay much more attention to word number 2 ("milk") than to the other two words

Attention Mechanisms



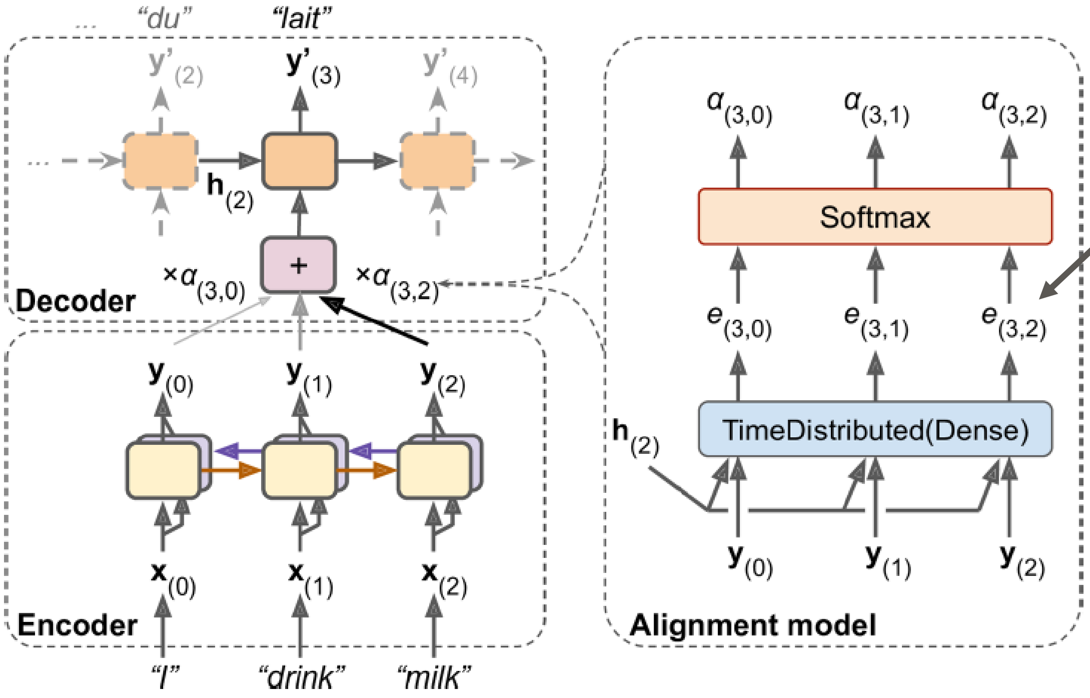
These weights are generated by a type of small neural network called an alignment model (or an **attention layer**), which is trained jointly with the rest of the Encoder–Decoder model

Attention Mechanisms



It starts with a time-distributed **Dense** layer with a single neuron, which receives as input all the encoder outputs, concatenated with the decoder's previous hidden state $h_{(2)}$

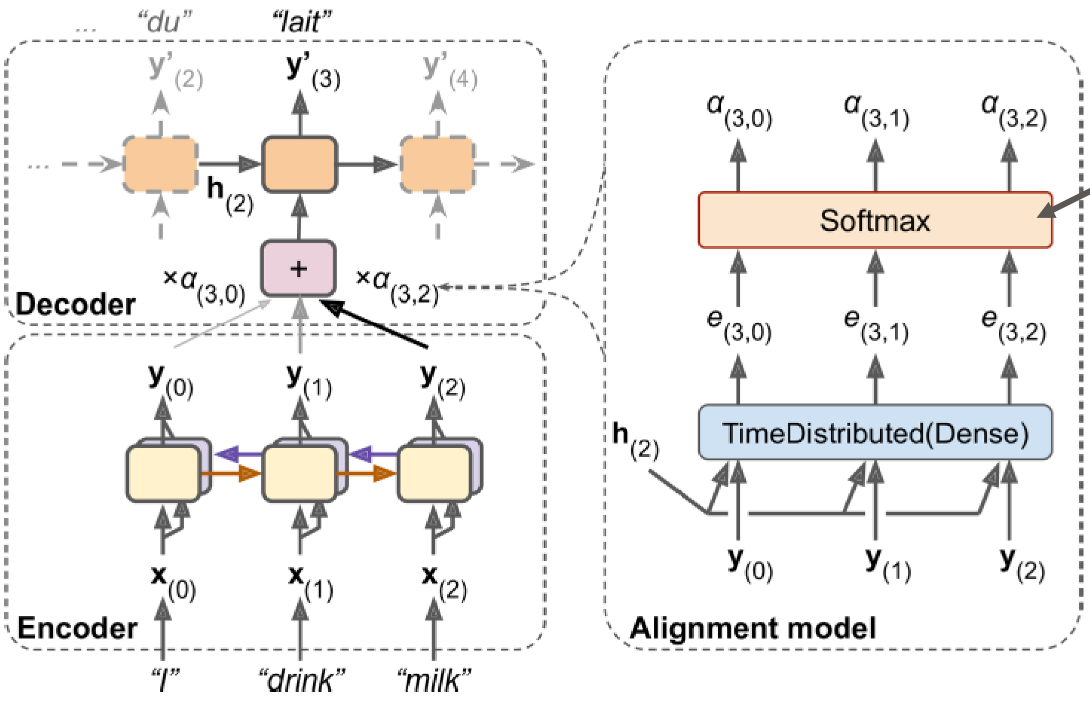
Attention Mechanisms



This layer outputs a score (or energy) for each encoder output $e_{(3,\ 2)}$

This score measures how well each output is aligned with the decoder's previous hidden state

Attention Mechanisms



Finally, all scores go through softmax layer to get final weight for each encoder output $\alpha_{(3,2)}$

All weights for a given decoder time step add up to 1 (since the softmax layer is not time-distributed)

Attention Mechanisms

- This attention mechanism is called **Bahdanau attention** which
- Concatenates encoder’s output with decoder’s previous hidden state
- So it is called **concatenative attention** (or additive attention)

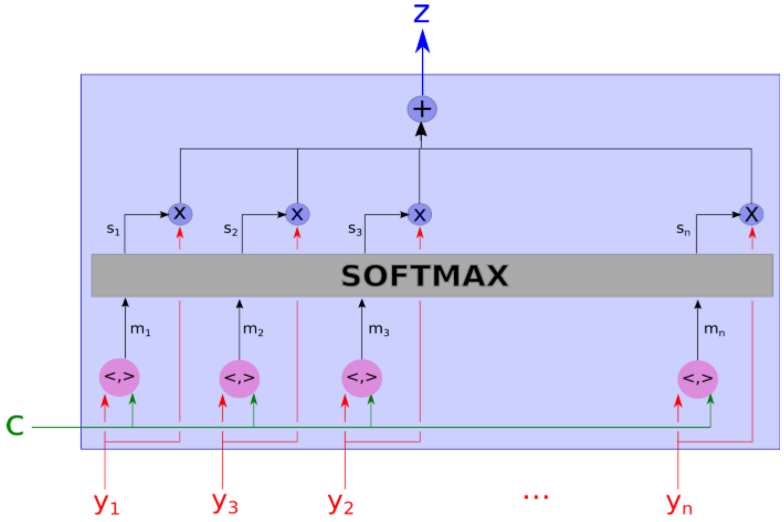
Types of Attention

- Global Attention
- Local Attention
- Hard Attention
- Soft Attention

Attention Mechanisms

- Another common attention mechanism proposed in 2015 by Minh-Thang Luong et al.
- It's called **Luong attention** or **multiplicative attention**.

$m = c.y$



Attention Mechanisms

Benefits of attention mechanisms:

- Significant improvement for long sentences
- Make it easier to understand what led the model to produce its output
    - This is called **explainability**