Session 11 - GRU, Attention Mechanism & Transformers: Attention is all you need!

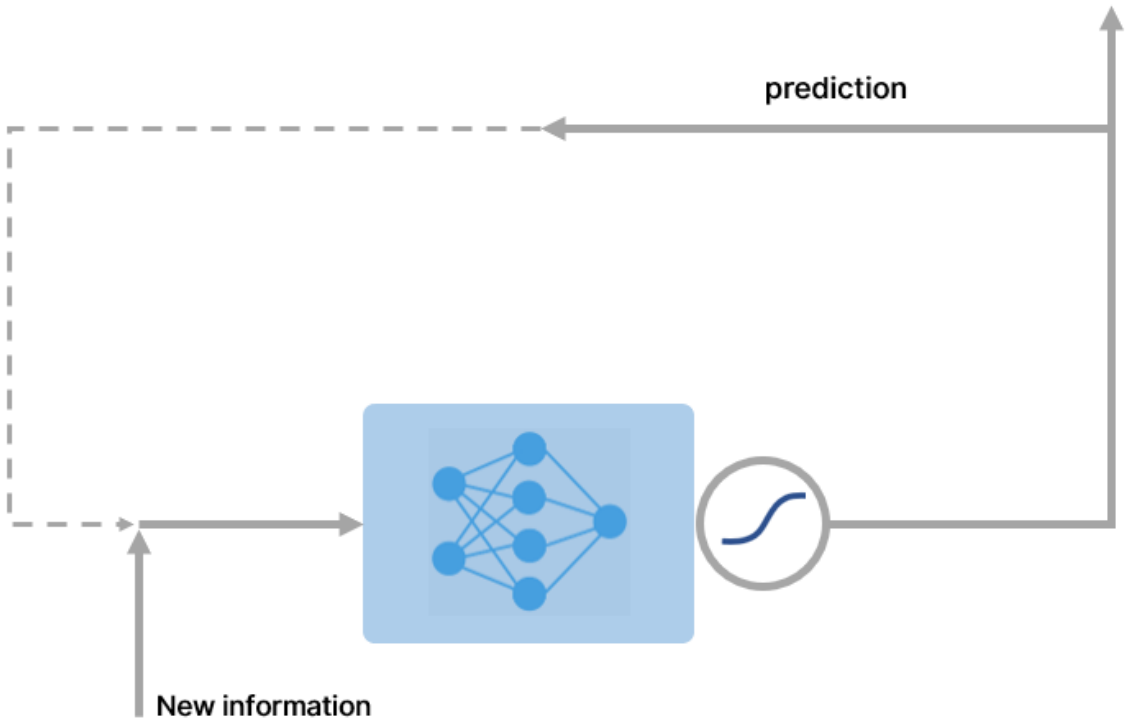
Submit Assignment

Due Saturday by 11:59pm **Points** 1,000 **Submitting** a text entry box or a website url
Available Oct 24 at 9am - Oct 31 at 11:59pm 8 days

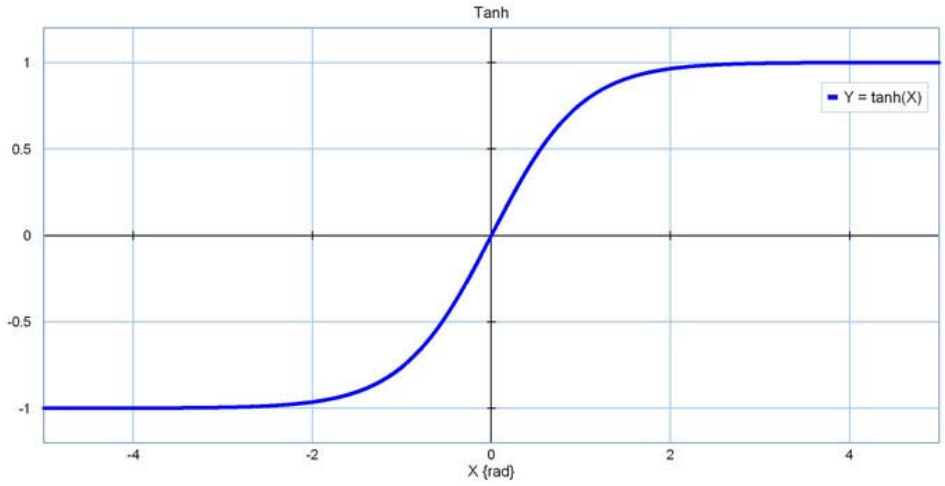
Session 11 - GRU, Attention Mechanism & Transformers

Attention is all you need!

RNN/LSTM Recap

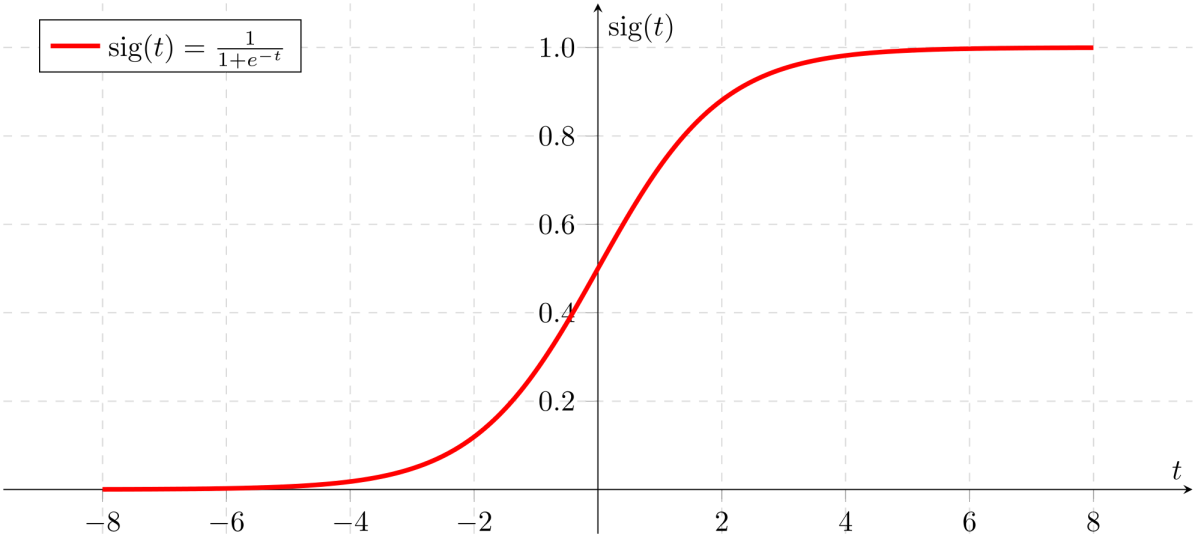


TanH - Squashing Function



Helps maintain output between -1 & +1. If we don't have it, then after 500 iterations a big output may explode!

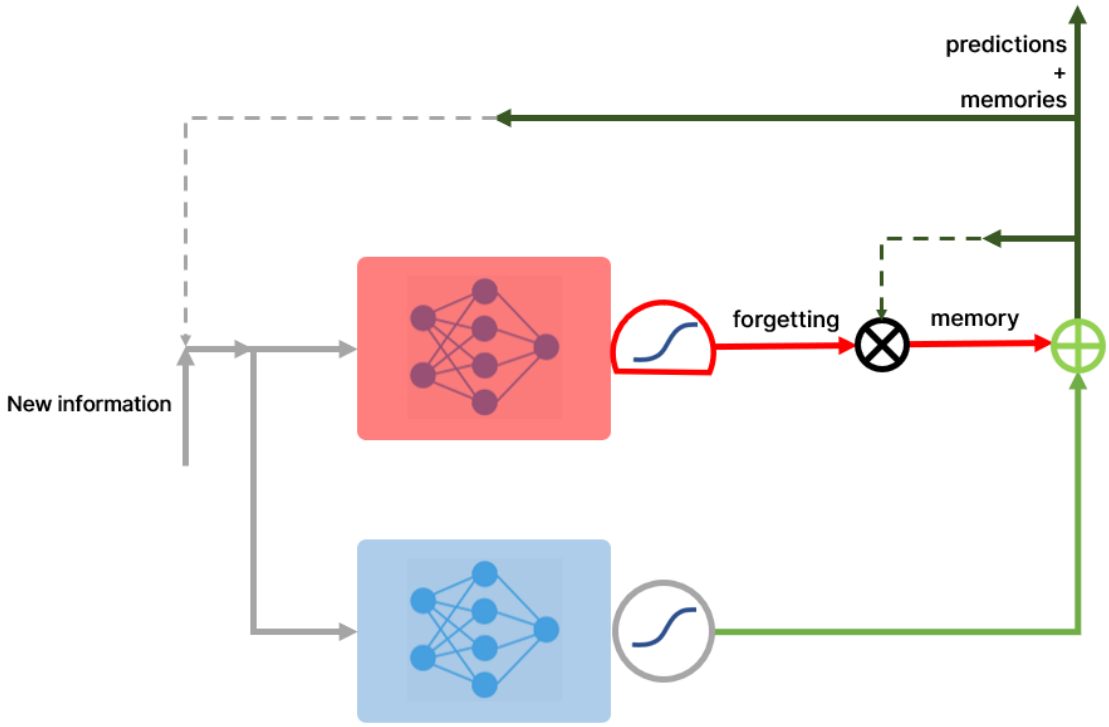
Sigmoid - Selection Gate



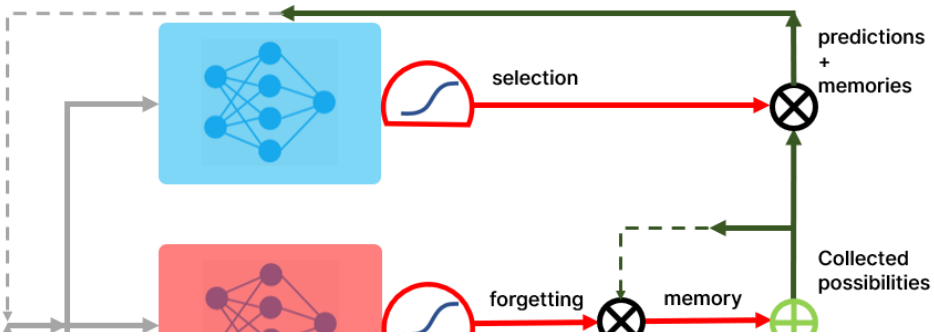
We'd like to use a function which is between 0 and 1 for this. We use sigmoid for that.

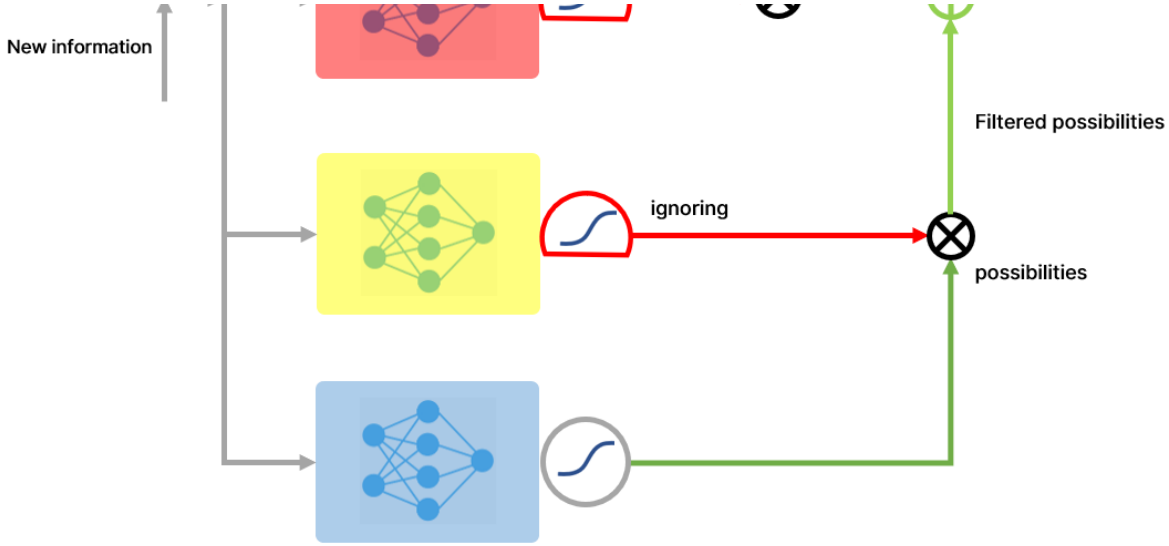
Elementwise multiplication acts like a router, controlling what goes out and what doesn't.

Adding Memory to our Network

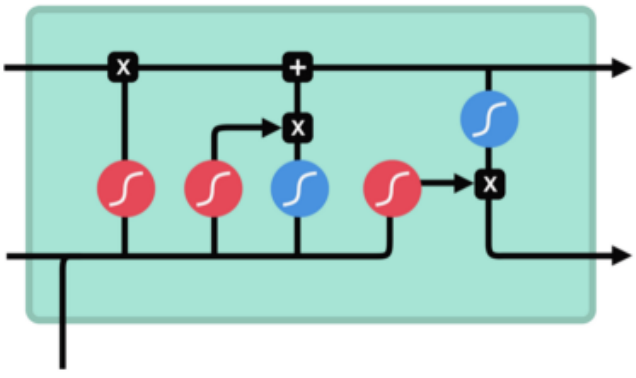


LSTMs

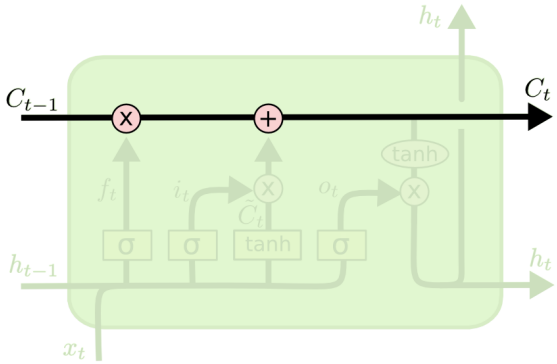




An LSTM has a similar control flow as a recurrent neural network. It processes data passing on information as it propagates forward. The differences are the operations within the LSTM's cells.



LSTM Internals



The key to LSTMs is the cell state or memory. It is shown as the horizontal line running through the top of the diagram (and as the circular loop in our image).

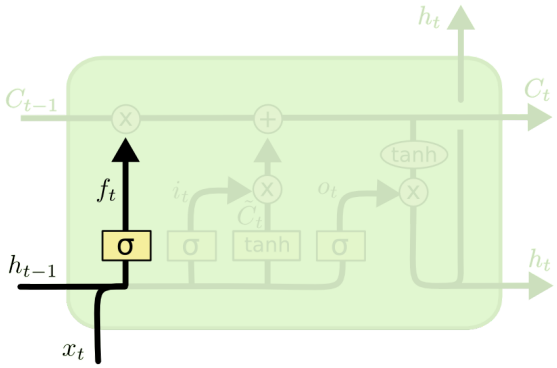
The cell-state or the memory is maintained by the forgetting gate. Unlike in RNN where will forget because of over-writes, memory in LSTM is maintained by the **forget gate** removing the un-necessary edits.

***Source (https://colah.github.io/posts/2015-08-Understanding-LSTMs/)** : You cannot get a better source than COLAH's blog on LMST/GRU etc.*

Let's go through LSTM's again step-by-step:

STEP 1 - Forget Gate

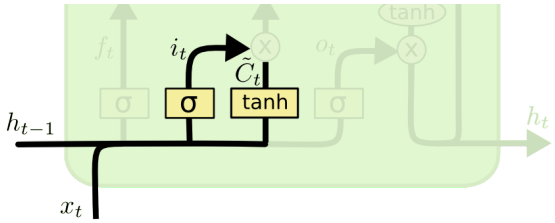
Forget the additional information which might have entered in the immediate last step, and maintain the long term information required. This is why it is called Forget Gate:



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

STEP 2 - Input Gate

Not let's decide what information we want to add based on the new-input. For this, we will use 1 DNN to predict all possible values, and other **re-scores** or **filter-outs** the values, like a manager, would.

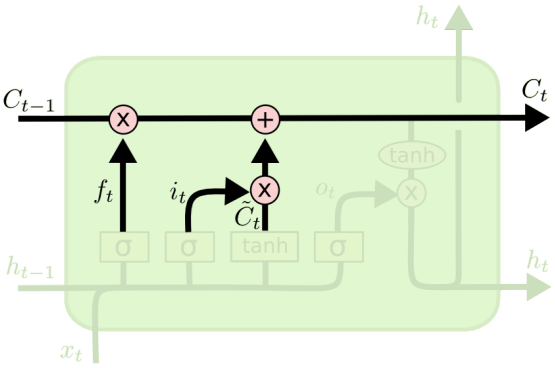


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

STEP 3 - Update the Memory/Context/Cell-State

Now let's forget whatever we might have added in the last step, and add new information through the input gate.

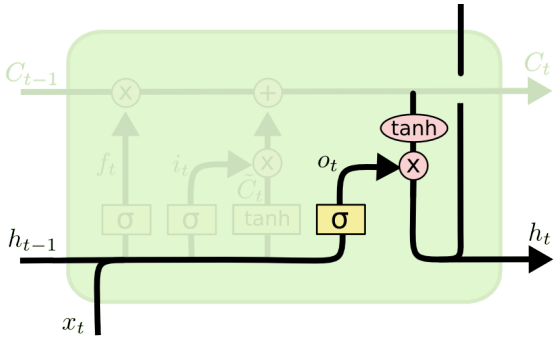


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

STEP 4 - The Output

Till now we have updated our Memory/Cell-state. It is time to now provide the output. Our Memory is not our output, we need to filter things out.

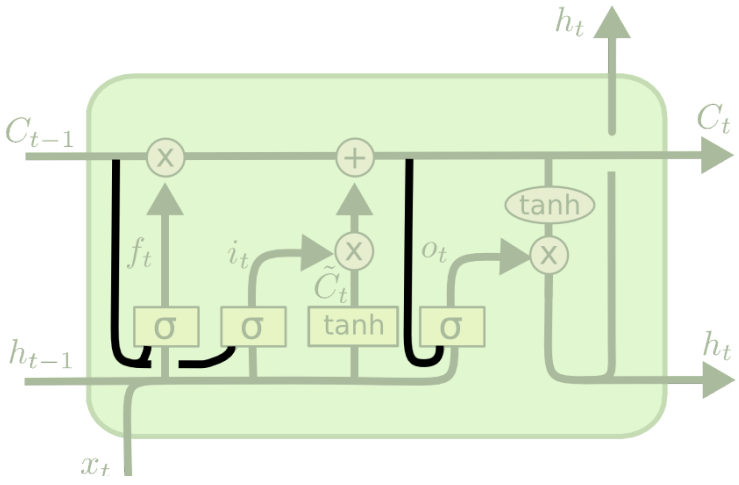
h_t



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

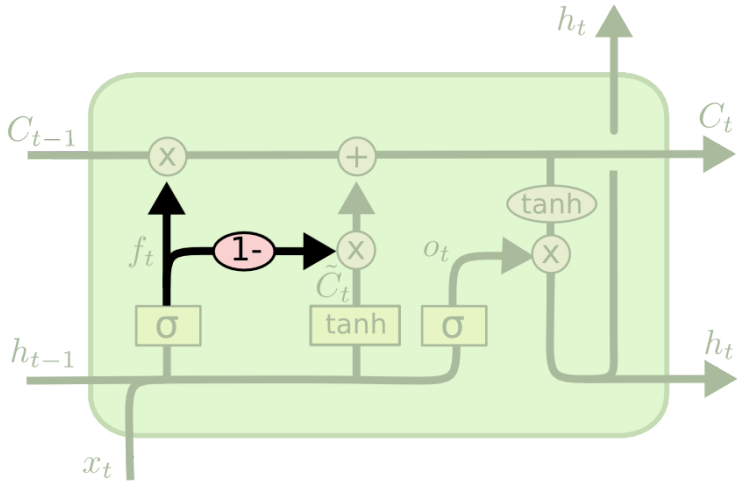
LSTM Variants

Peephole: Now since you have gone through simple LSTM, we can create many different variants. Look at this one called Peephole LSTM.



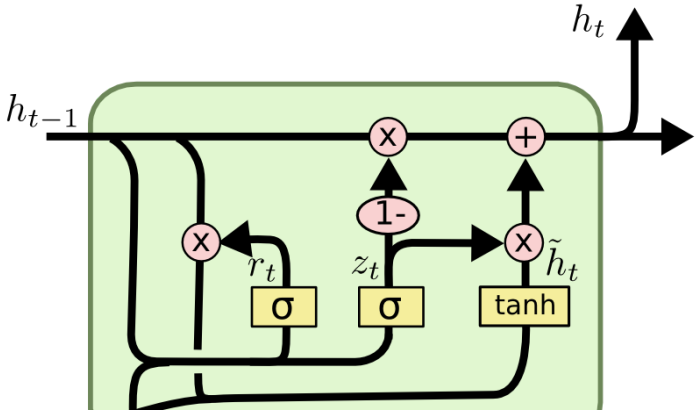
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t\right] -$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t\right] -$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t\right] +$$

Coupled Forget-Input LSTM: Instead of separately deciding what to forget and what we should add, we can make those decisions together. We only forget when we're going to input something in its place. Forget and Input becomes Ying-Yang!



$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) *$$

GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t\right]$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}$$

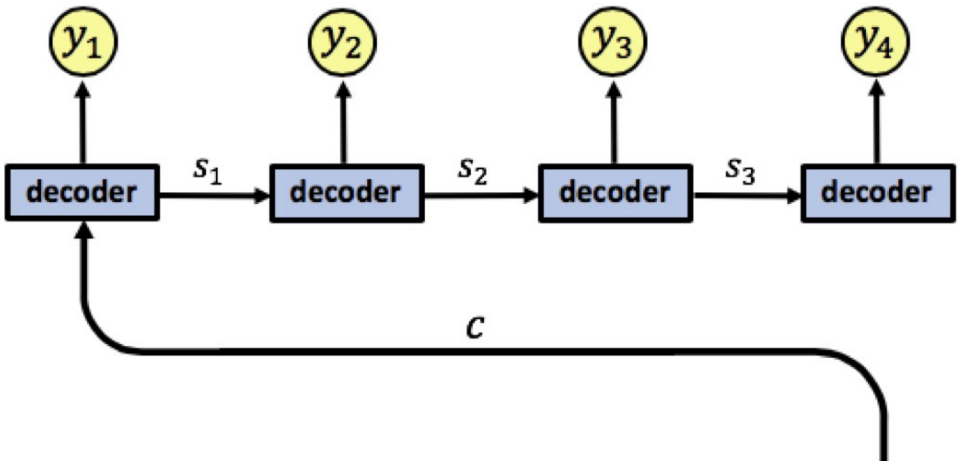
A slightly more dramatic variant on the LSTM is the **Gater Recurrent Unit** or GRU.

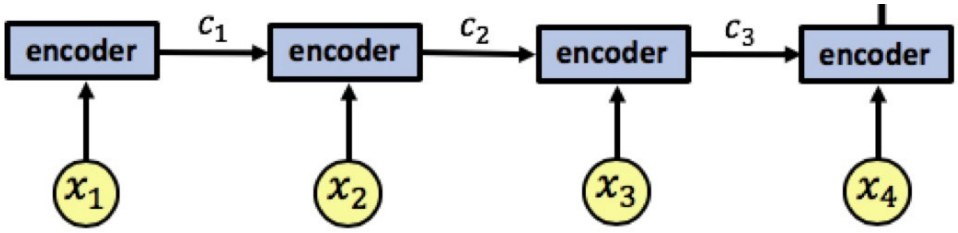
- It combines the forget and input gates into a single "update gate".
- It also merges the cell state and hidden state
- It updates the memory twice, the first time (using old state and new input, called **Reset Gate**) and the second time (as final output).
- Old cell state or hidden state (with input) is used for its own update as well as for deciding

GRU has been growing increasingly popular and is a default alternative for LSTMs.

Encoder-Decoder Architecture in RNN/LSTMs

The RNN encoder-decoder architecture looks like this [source (https://medium.com/datadriveninvestor/attention-in-rnns-321fbcd64f05)]



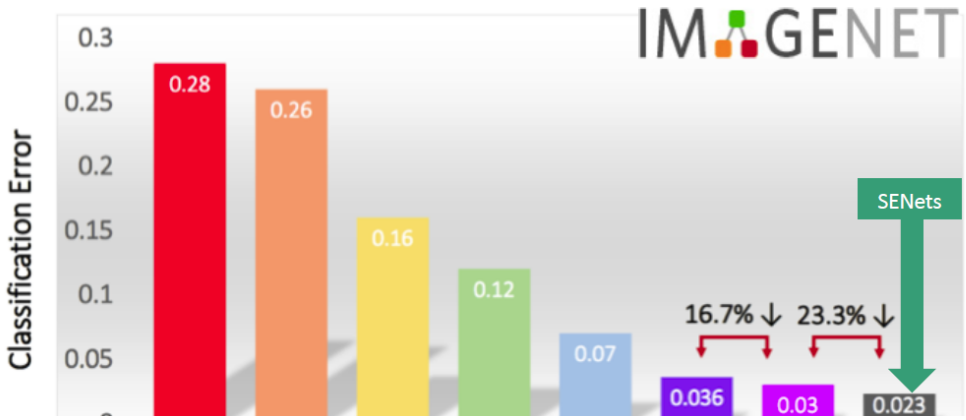


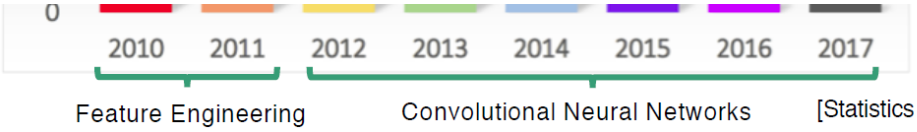
The RNN encoder has an input sequence x1, x2, x3, x4. We denote the encoder states by c1, c2, c3. The encoder outputs a single output vector c which is passed as input to the decoder. Like the encoder, the decoder is also a single-layered RNN, we denote the decoder states by s1, s2, s3 and the network's output by y1, y2, y3, y4.

A problem with this architecture lies in the fact that the decoder needs to represent the entire input sequence x1, x2, x3, x4 as a single vector c, which can cause information loss. Moreover, the decoder needs to decipher the passed information from this single vector, a complex task in itself.

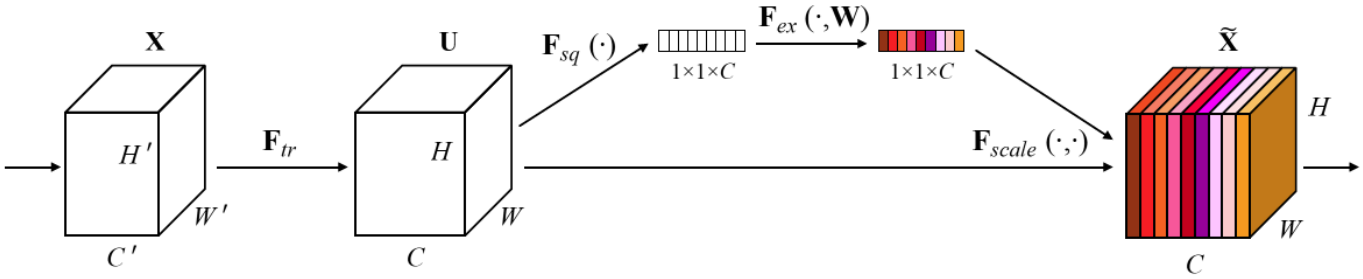
RNN/LSTM with an Attention Mechanism

First let's take a look at a variant of ResNet called **SENet** **(https://arxiv.org/pdf/1709.01507.pdf)** .

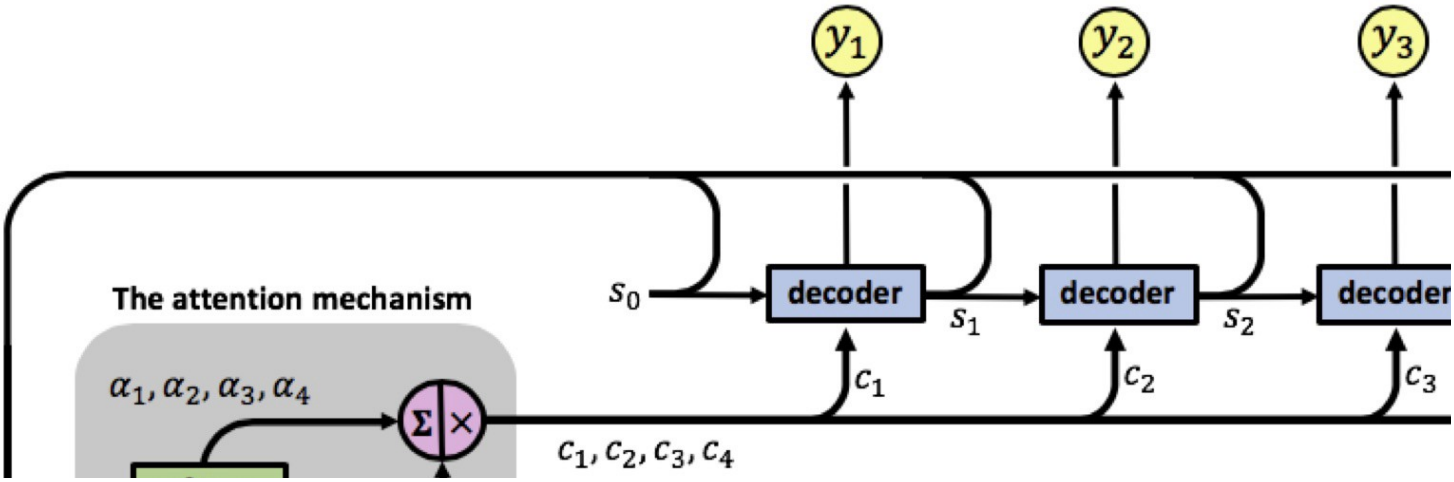


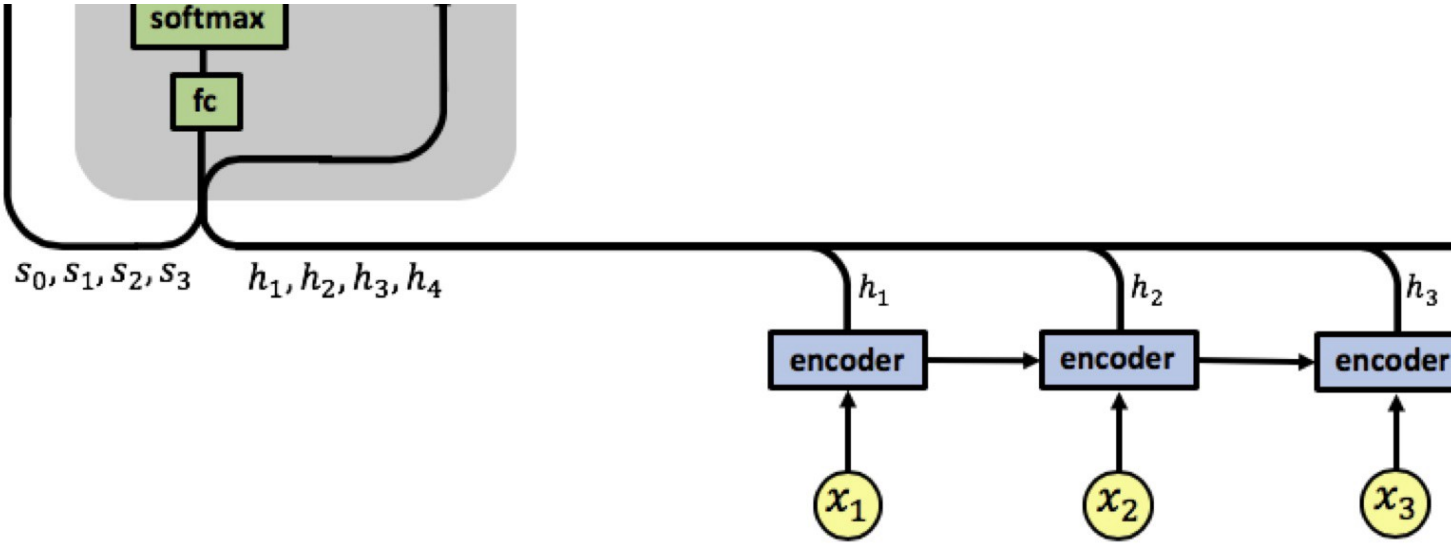


Squeeze and Excitation Block



An attention RNN/LSTM looks like this:





Our attention model has a single RNN encoder, again with 4-time steps. We denote the encoder's input vectors by $x_1,\ x_2,\ x_3,\ x_4$ and the output vectors by $h_1,\ h_2,\ h_3,\ h_4$.

The attention mechanism is located between the encoder and the decoder.

Its output is composed of the encoder's input vectors $h_1,\ h_2,\ h_3,\ h_4$ and the states of the decoder $s_0,\ s_1,\ s_2,\ s_3$, the attention's output is a sequence of vectors called **context vectors** denoted by $c_1,\ c_2, c_3, c_4$.

The Context Vector

The context vectors enable the decoder to focus on certain parts of the input when predicting its output.

Each context vector is a weighted sum of the encoder's output vector $h_1,\ h_2,\ h_3,\ h_4$

Each vector h_i contains information about the **whole input sequence up to that moment with a strong focus on the i^{th} stage**.

The vectors $h_1,\ h_2,\ h_3,\ h_4$ are scaled by weights α_{ij} capturing the degree of relevance of input.

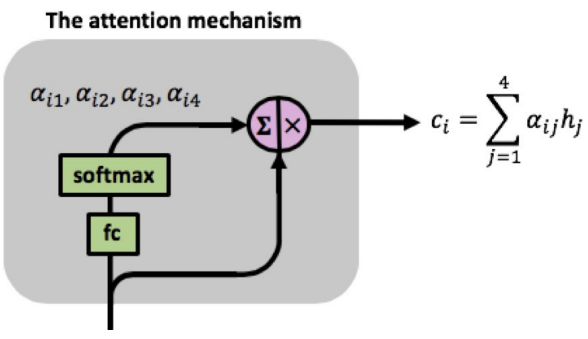
The context vectors $c_1,\ c_2,\ c_3,\ c_4$ are given by:

$$c_{ij} = \sum_{j=1}^{4} \alpha_{ij} h_j$$

The attention weights are learned using an additional fully connected shallow network, denoted by **fc**, this is where the $s_0,\ s_1,\ s_2,\ s_3$ part of the attention mechanism's input comes into play. Computation of the attention weights are given by:

$$\alpha_{ij} = \frac{exp(fc(s_{i-1},\ h_j))}{\sum_{k=1}^{4} fc(s_{i-1},\ h_k)}$$

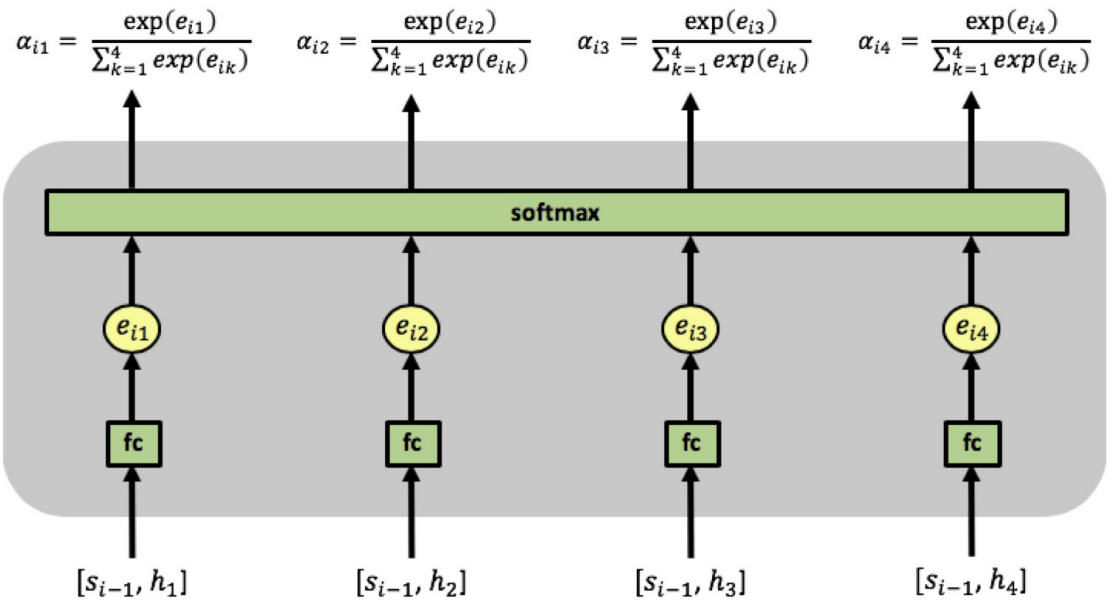
The attention weights are learned using the attention fully-connected network and a softmax function:



s_{i-1} h_1, h_2, h_3, h_4

As can be seen in the image above, the fc receives the concatenated vectors $[s_{i-1},\ h_i]$ as the input at time step i. The network has a single **fc** layer, the outputs of the layer are passed through a softmax function computing the attention weights

Notice that we are using the **same** fully-connected network for all the concatenated pairs $[s_0,\ h_1]$, $[s_0,\ h_2]$, $[s_0,\ h_3]$, $[s_0,\ h_4]$



The fc network is trained along with the encoder and decoder using backpropagation, the RNN's prediction error terms are backpropagated backward through the decoder, then through the fc attention network and from there to the encoder.

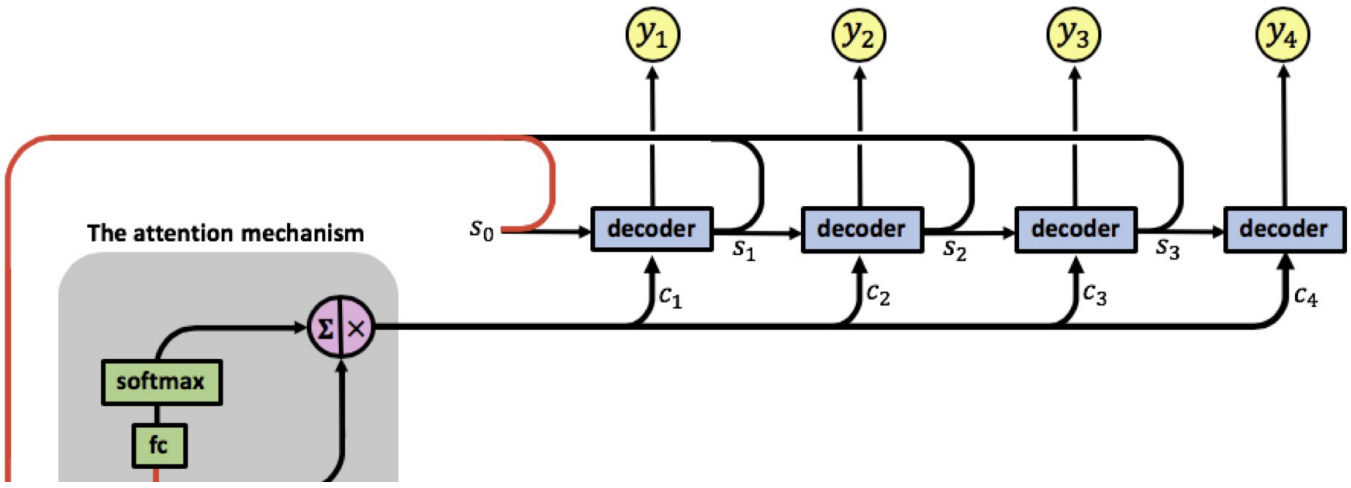
By letting the decoder have an attention mechanism, we relieve the encoder from having to encode all information in the input sequence into a single vector.

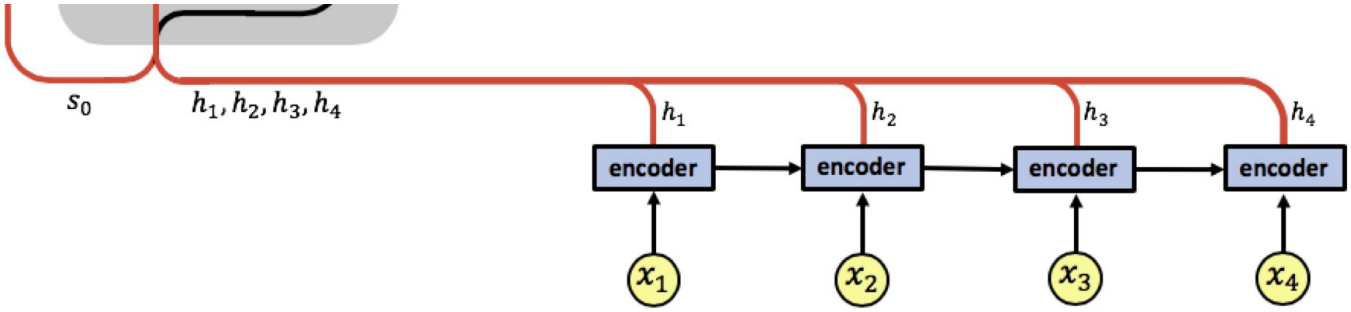
Computing Attention Weights and Context Vectors

Let's go over a detailed example:

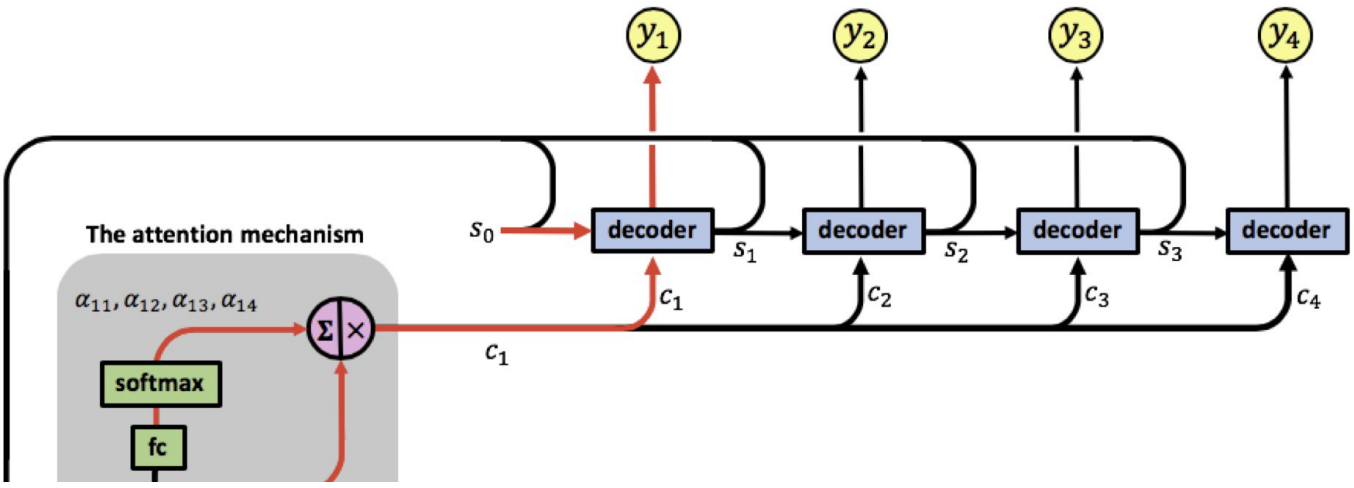
The first act performed is the computation of vectors $h_1,\ h_2,\ h_3,\ h_4$ by the encoder.

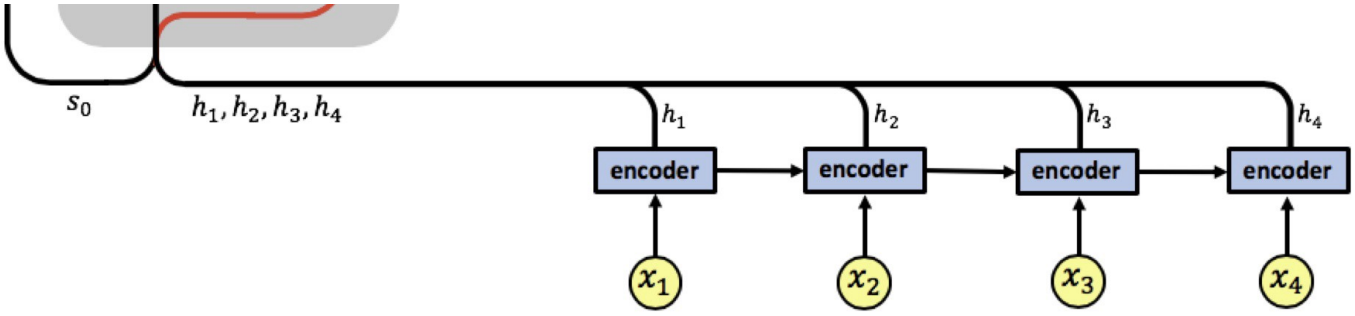
These are then used as inputs of the attention mechanism. This is where the decoder is first involved by inputting its initial state vector and we have the **first attention input sequence** $[s_0,\ h_1]\ ,\ [s_0,\ h_2]\ ,\ [s_0,\ h_3]\ ,\ [s_0,\ h_4]$





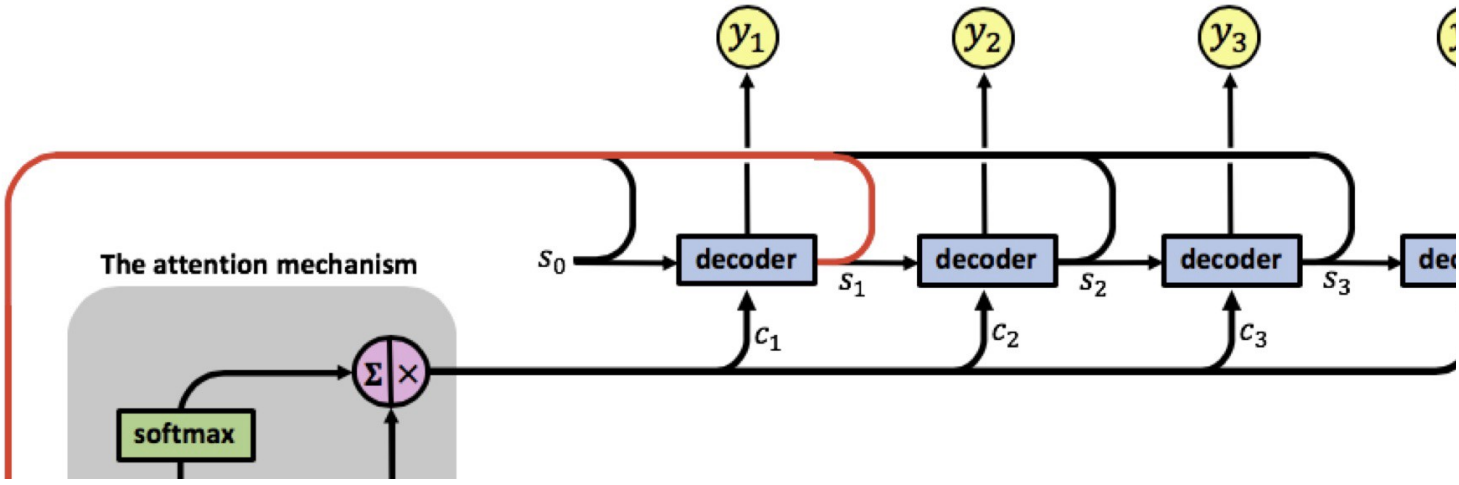
he attention mechanism computes the first set of attention weights enabling the computation of the first context vector c_1. The decoder now uses $[s_0, c_1]$ and computes the first RNN output \(y_1\).

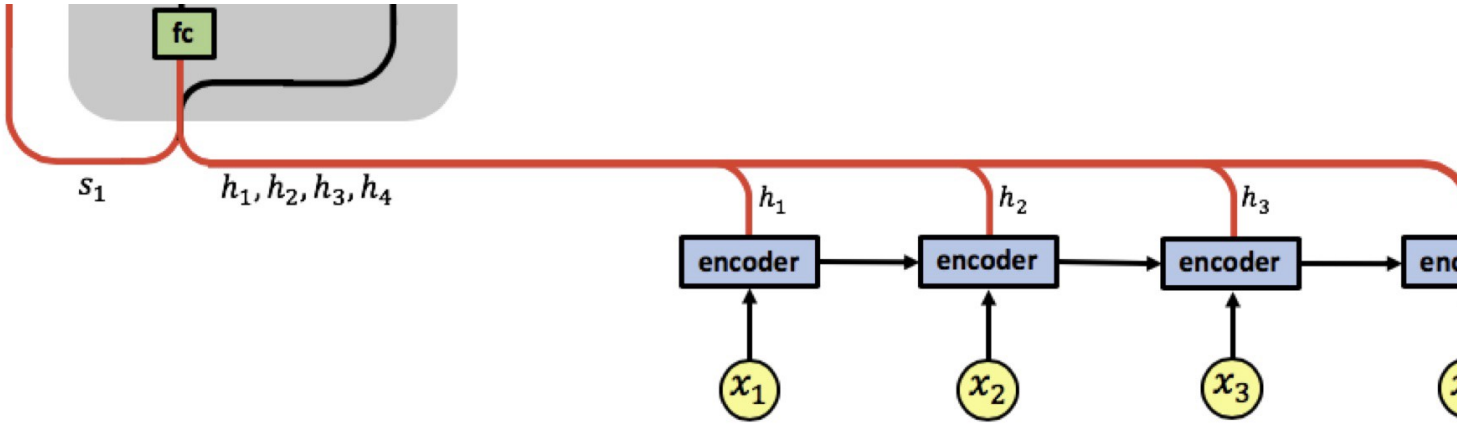




At the following step, the attention mechanism has as input the sequence

$$[s_1,\ h_1]\ ,\ [s_1,\ h_2]\ ,\ [s_1,\ h_3]\ ,\ [s_1,\ h_4]$$



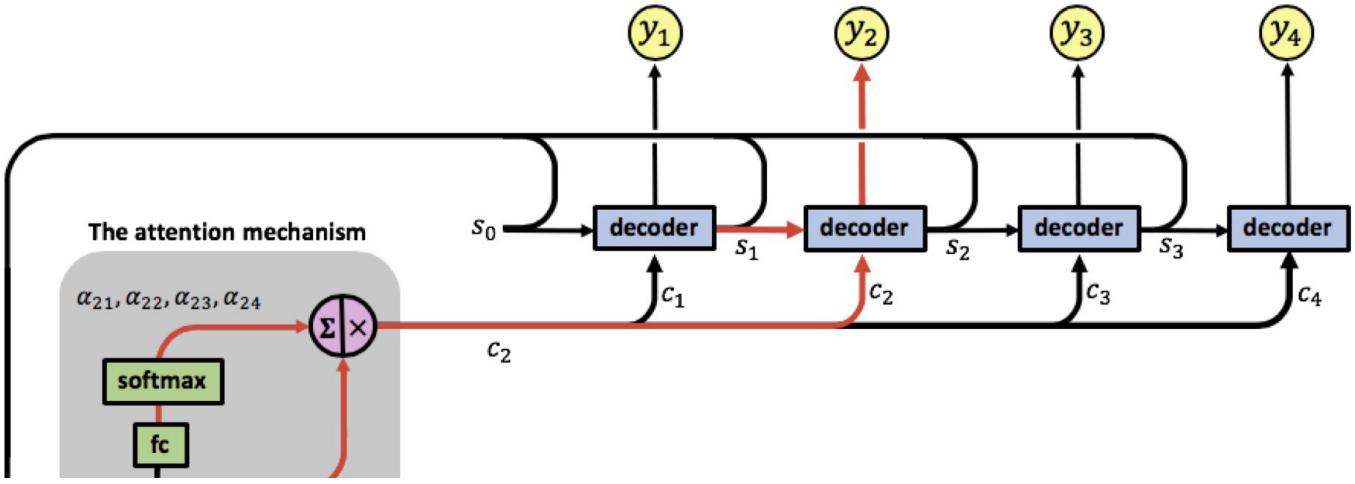


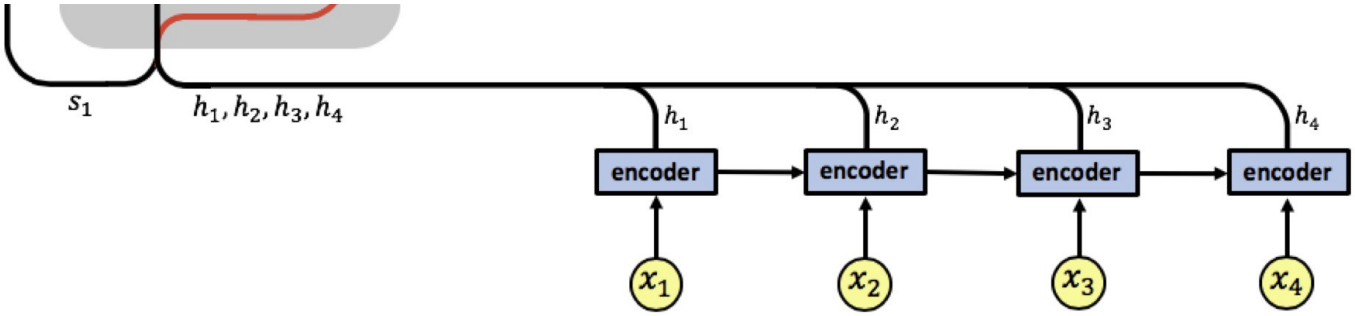
It computes a second set of weights enabling computation of the second context vector c_2. The decoder uses

$$[s_1, c_2]$$

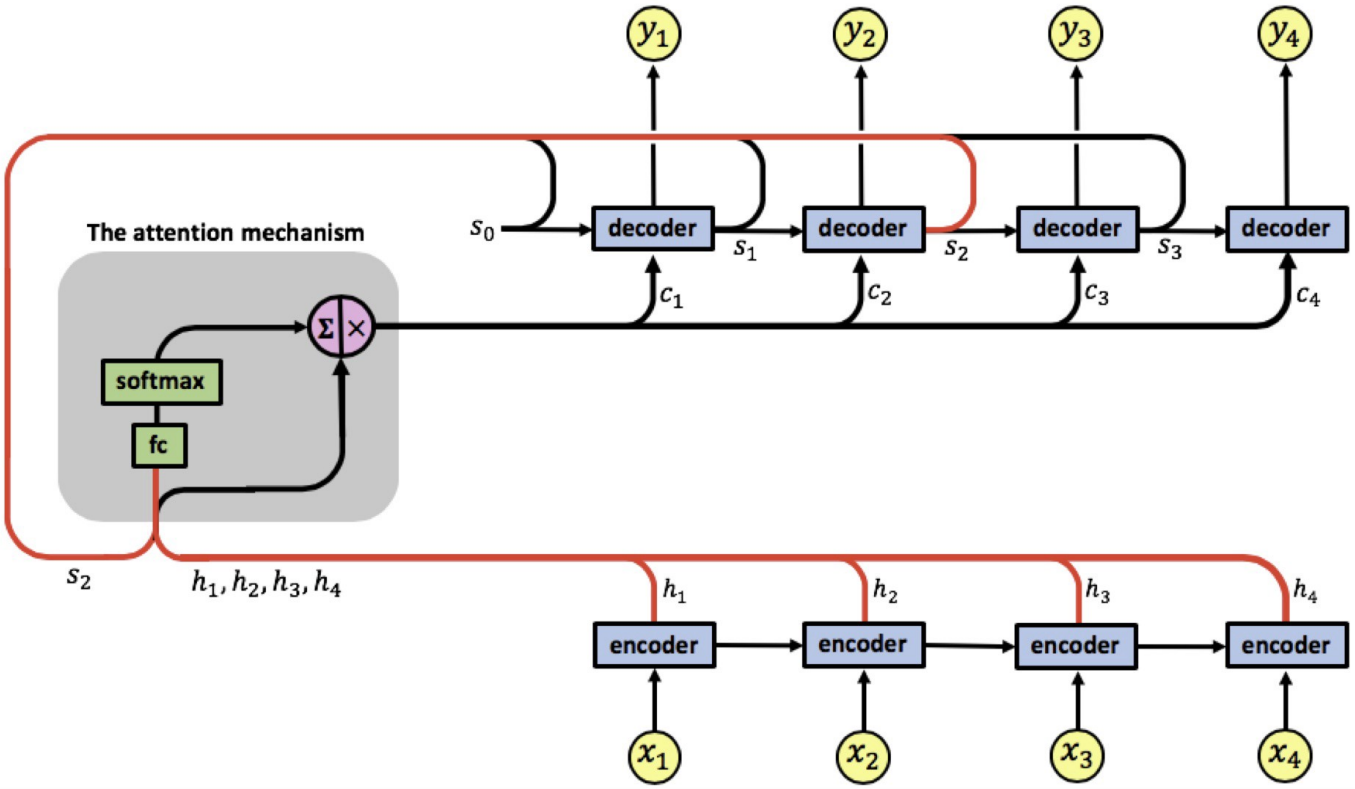
and computes the second RNN output

$$y_2.$$

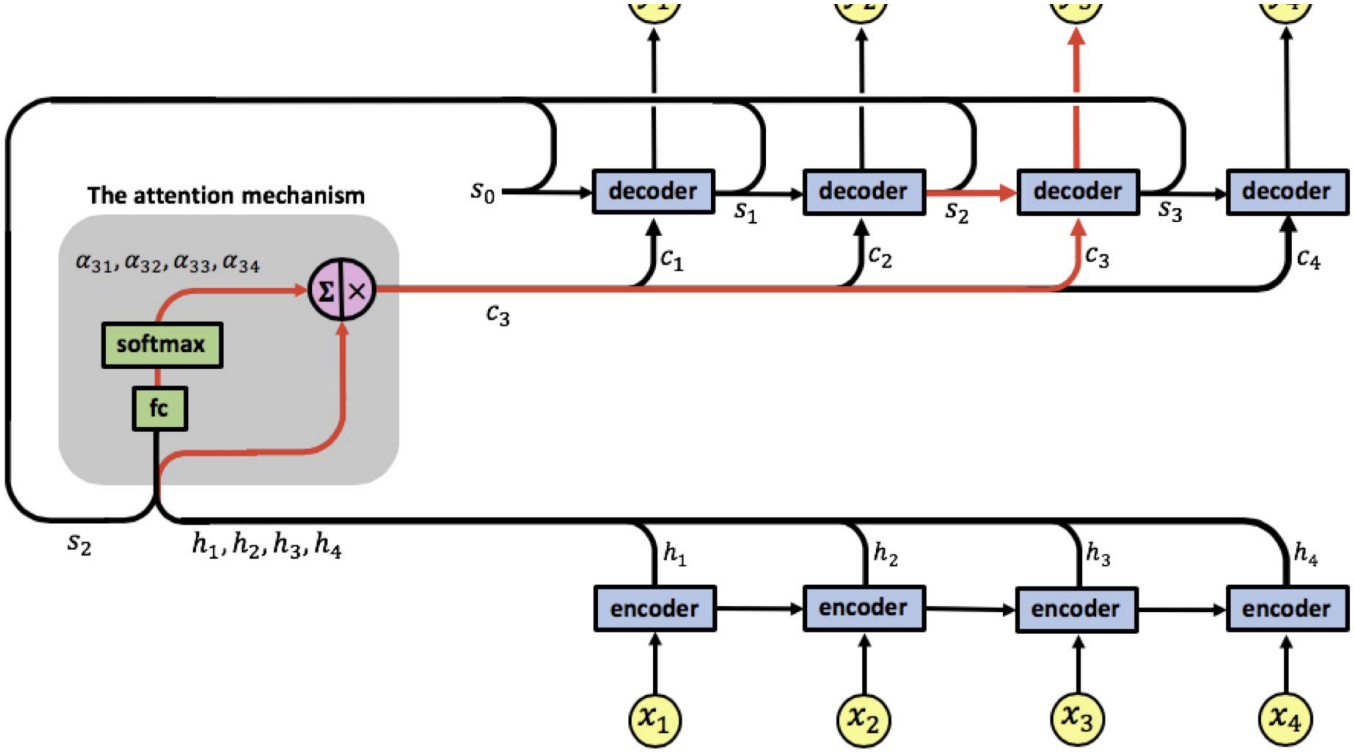




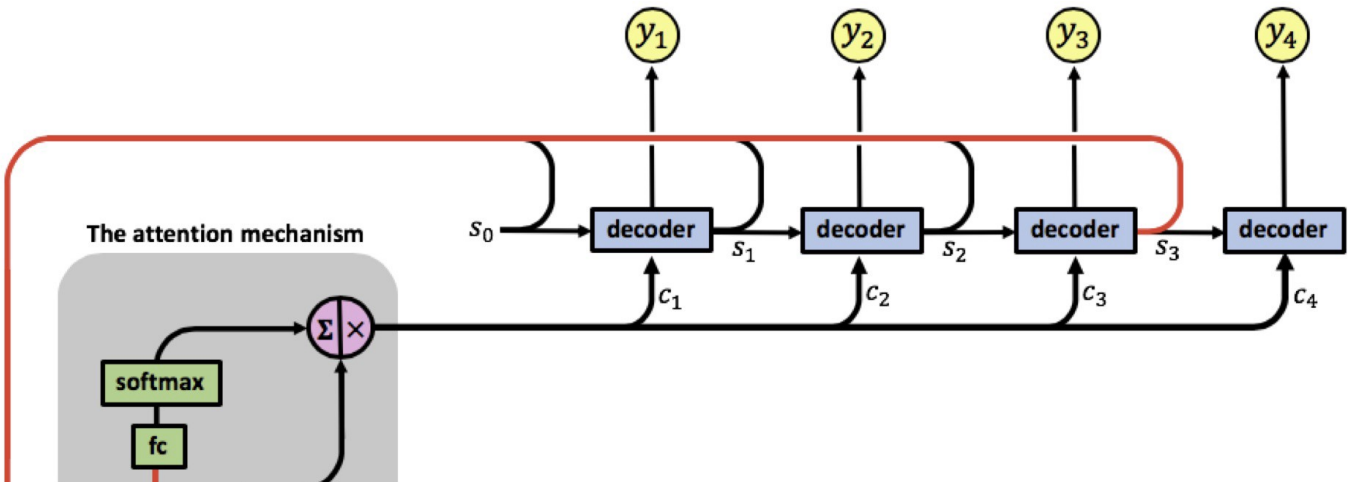
This next step should be clear:

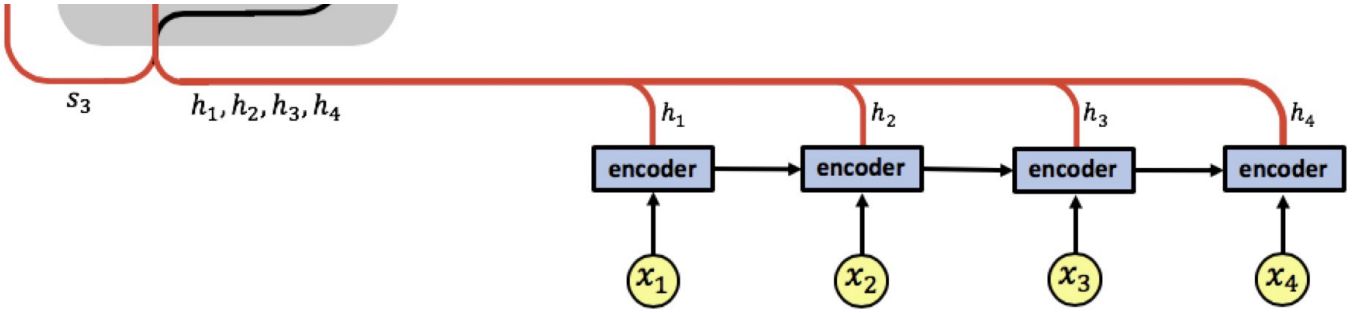


then this:

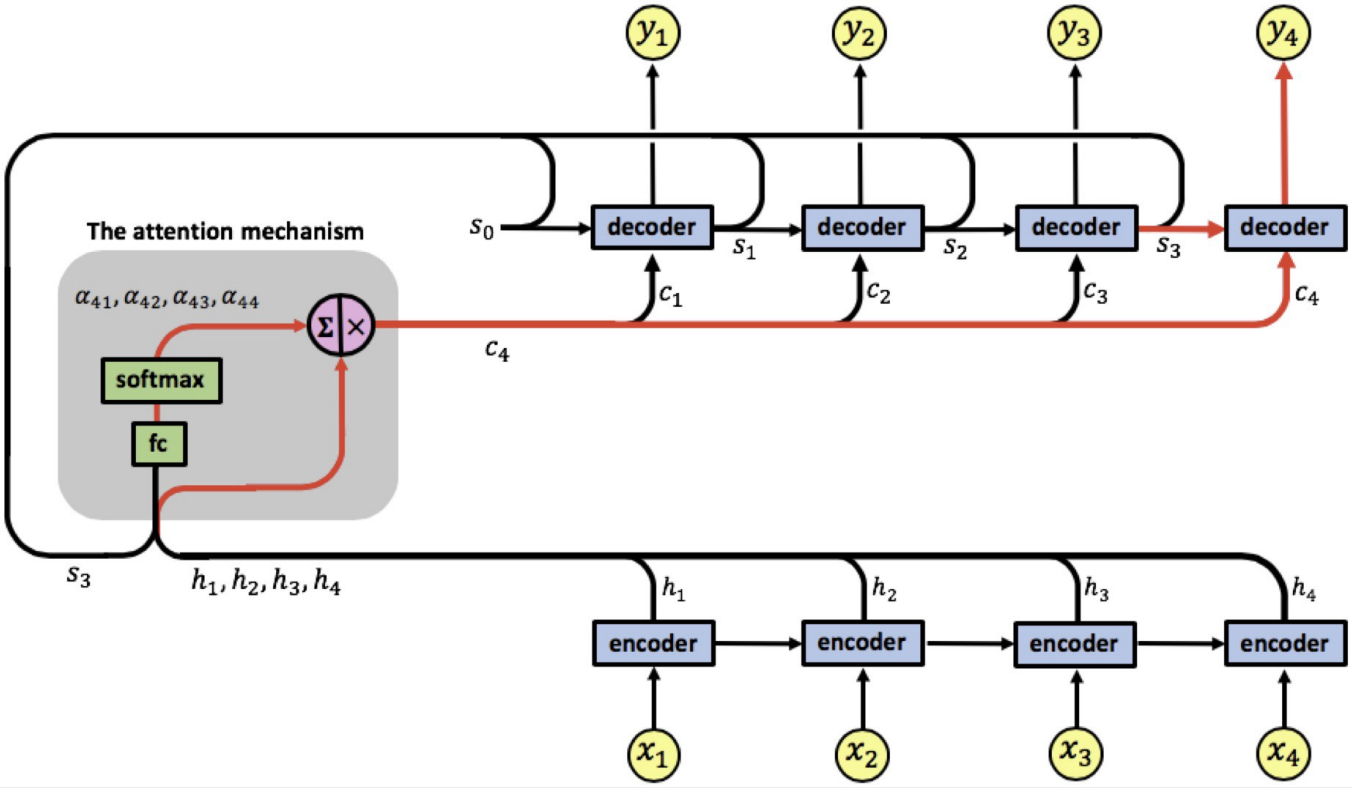

The attention mechanism
$\alpha_{31}, \alpha_{32}, \alpha_{33}, \alpha_{34}$
softmax
fc
s_0
s_1
s_2
s_3
c_1
c_2
c_3
c_4
c_3
s_2
h_1, h_2, h_3, h_4
h_1
h_2
h_3
h_4
decoder
encoder
x_1
x_2
x_3
x_4

Then this:


The attention mechanism
softmax
fc
s_0
s_1
s_2
s_3
c_1
c_2
c_3
c_4
y_1
y_2
y_3
y_4
decoder



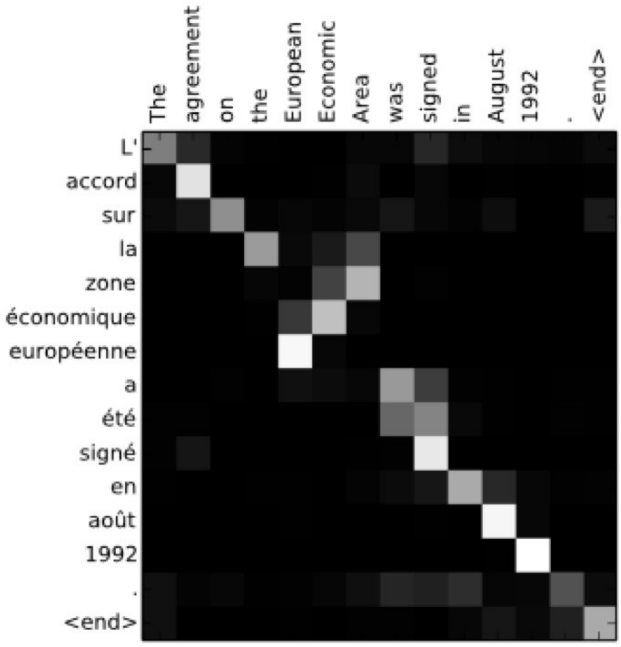
and finally this:

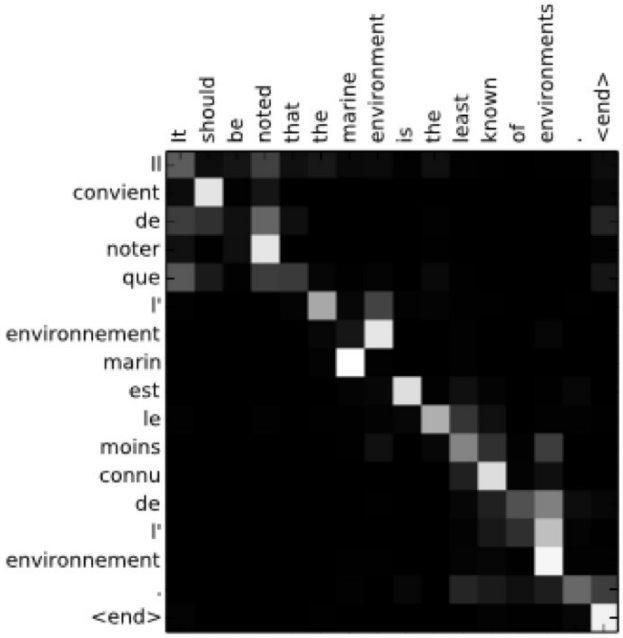


In the end, we have consumed 4 vectors, and with attention focused on a specific vector, we should and then made the prediction for each step.

Below are two alignments found by the attention RNN. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively.

Each pixel shows the weight α_{ij} of the j^{th} source word and the i^{th} target word, in grayscale (0: black, 1: white).





Better illustrations can be found here: [https://distill.pub/2016/augmented-rnns/ (https://distill.pub/2016/augmented-rnns/)](https://distill.pub/2016/augmented-rnns/)

Assignment

1. Look at this [file (https://bastings.github.io/annotated_encoder_decoder/)](https://bastings.github.io/annotated_encoder_decoder/)
2. Re-implement this and move to lambda. Provide the option to provide German text and get English text back.


EVA4P2S11