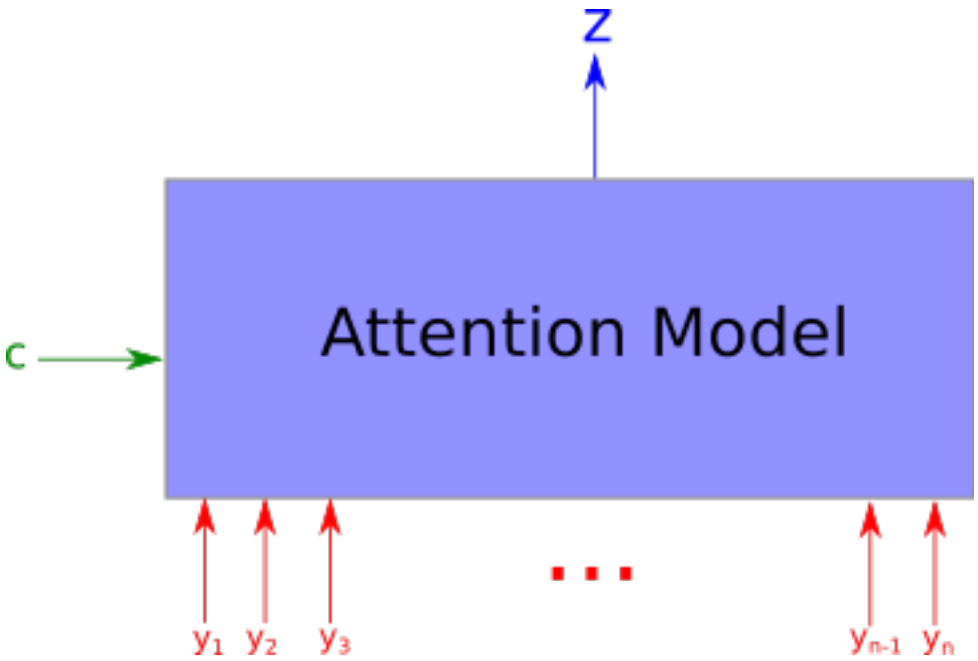
What is an attention model ?

What is an attention model, in a general setting ?

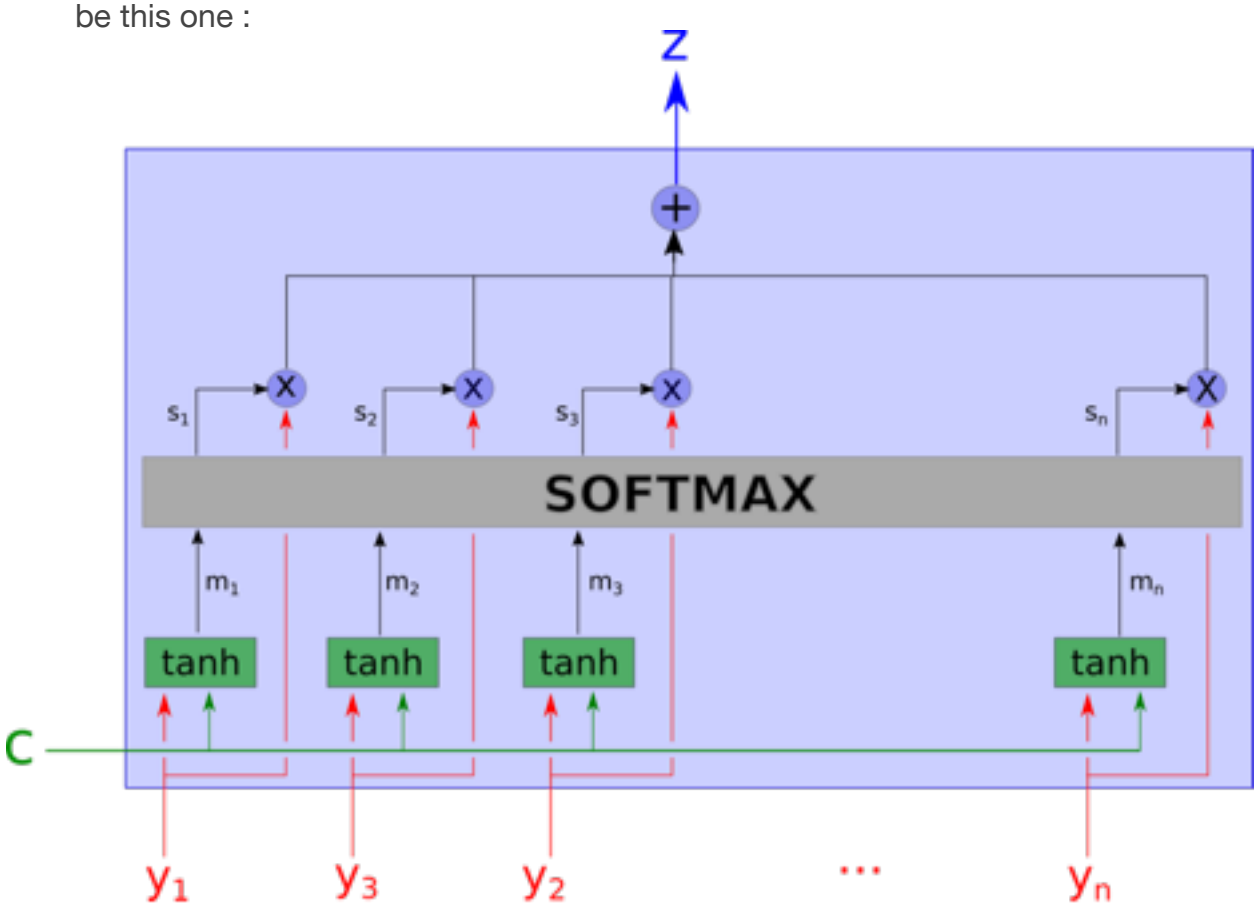


An attention model is a method that takes n arguments $y_1, \dots, y_n$ (in the precedent examples, the y_i would be the h_i), and a context c. It return a vector z which is supposed to be the « summary » of the y_i, focusing on information linked to the context c. More formally, it returns a weighted arithmetic mean of the y_i, and the weights are chosen according the relevance of each y_i given the context c.

In the example presented before, the context is the beginning of the generated sentence, the y_i are the representations of the parts of the image (h_i), and the output is a representation of the filtered image, with a filter putting the focus of the interesting part for the word currently generated.

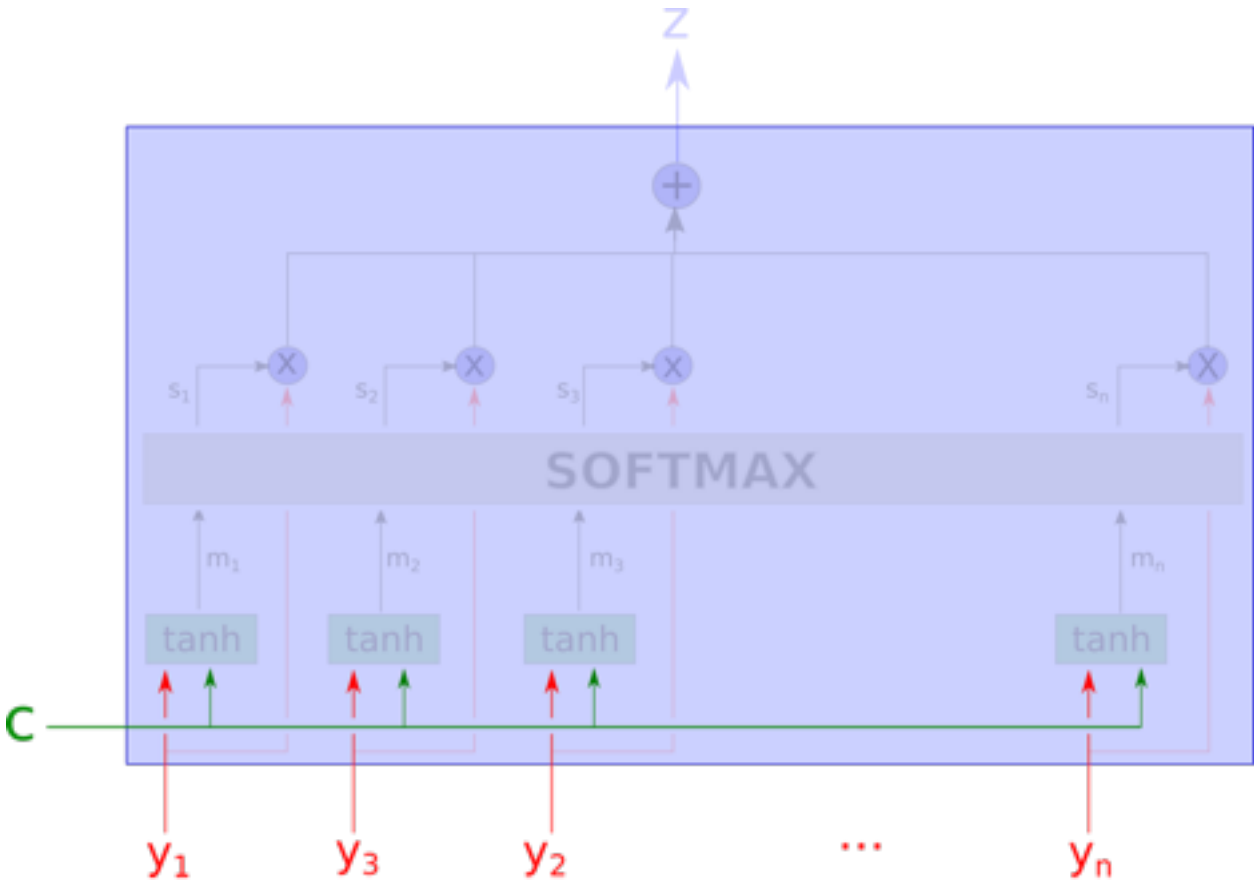
One interesting feature of attention model is that the weight of the arithmetic means are accessible and can be plotted. This is exactly the figures we were showing before, a pixel is whiter if the weight of this image is high.

But what is exactly this black box doing ? A figure for the whole attention model would be this one :

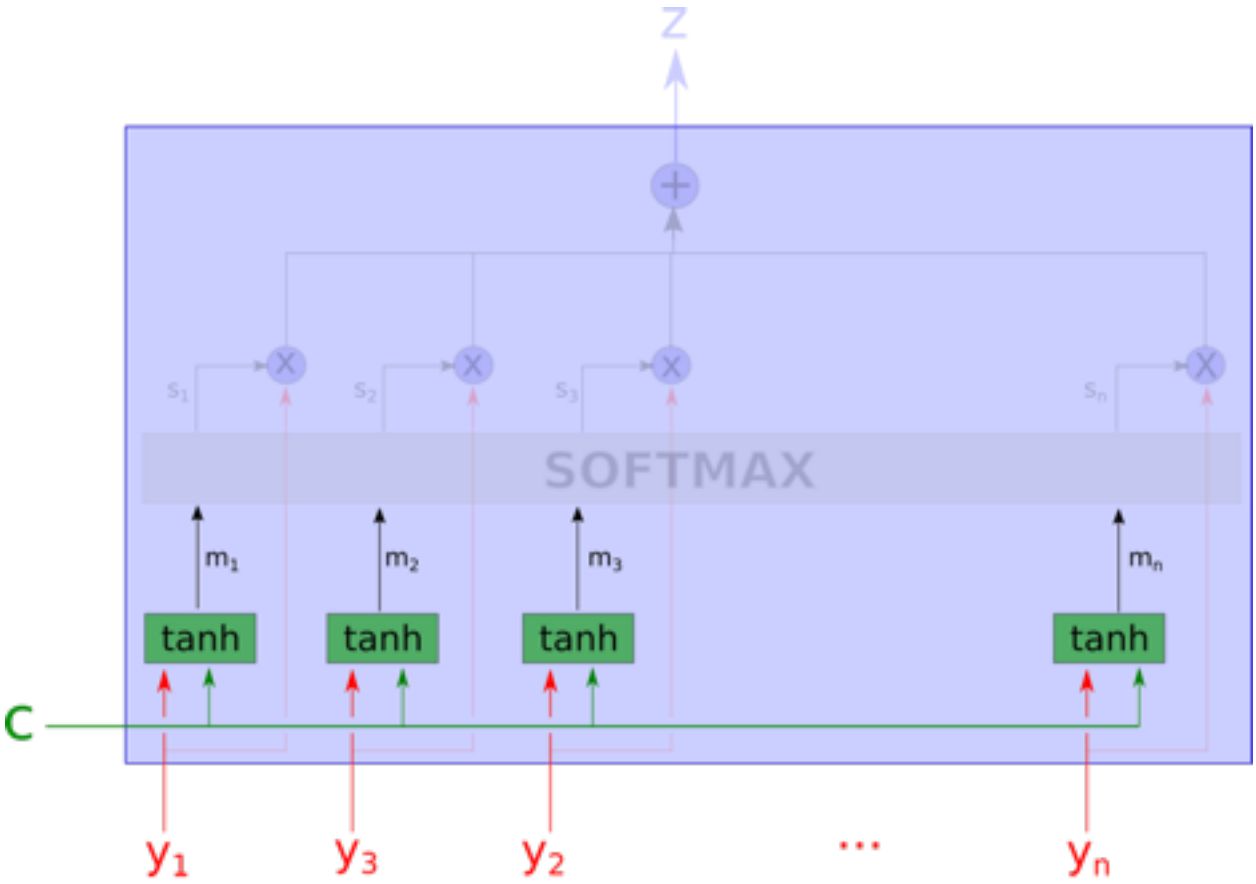


This network could seem to be complicated, but we are going to explain it step by step.

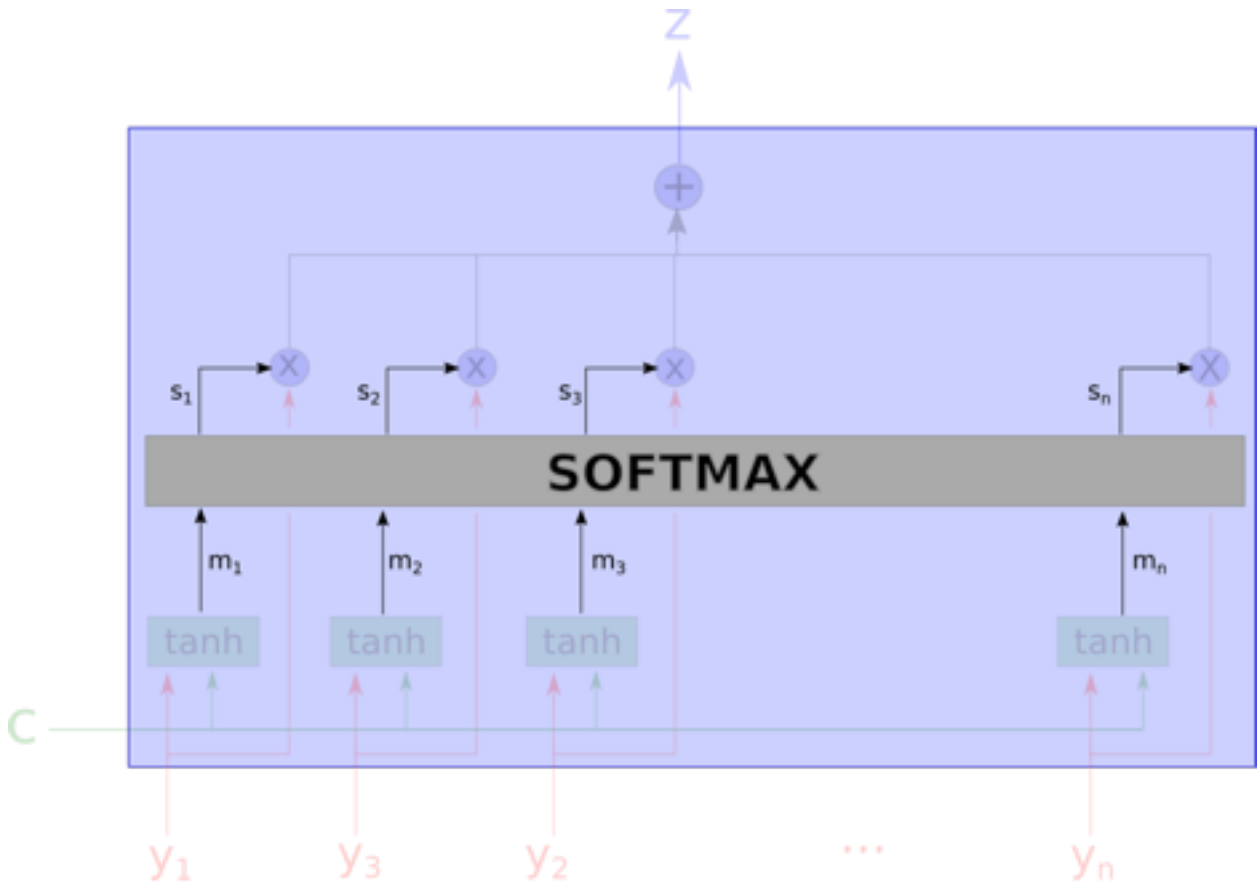
First, we recognize the input. c is the context, and the y_i are the « part of the data » we are looking at.



At the next step, the network computes m_1, … m_n with a $\tanh$ layer. It means that we compute an « aggregation » of the values of y_i and c. An important remark here is that each m_i is computed without looking at the other y_j for $j \neq i$. They are computed independently.



Then, we compute each weight using a softmax. The softmax, as it name says, behaves almost like a argmax, but is differentiable. Let's say that we have an argma$ function such that $argmax(x_1, ..., x_n) = (0, ..., 0, 1, 0, ..., 0)$ where the only 1 in the output is telling which input is the max. Then, the softmax is defined by $softmax(x_1, ..., x_n) = (\frac{e^{x_i}}{\sum_j e^{x_j}})_i$. If one of the x_i is bigger than the other, then $softmax(x_1, ..., x_n)$ will be very close to $argmax(x_1, ..., x_n)$.

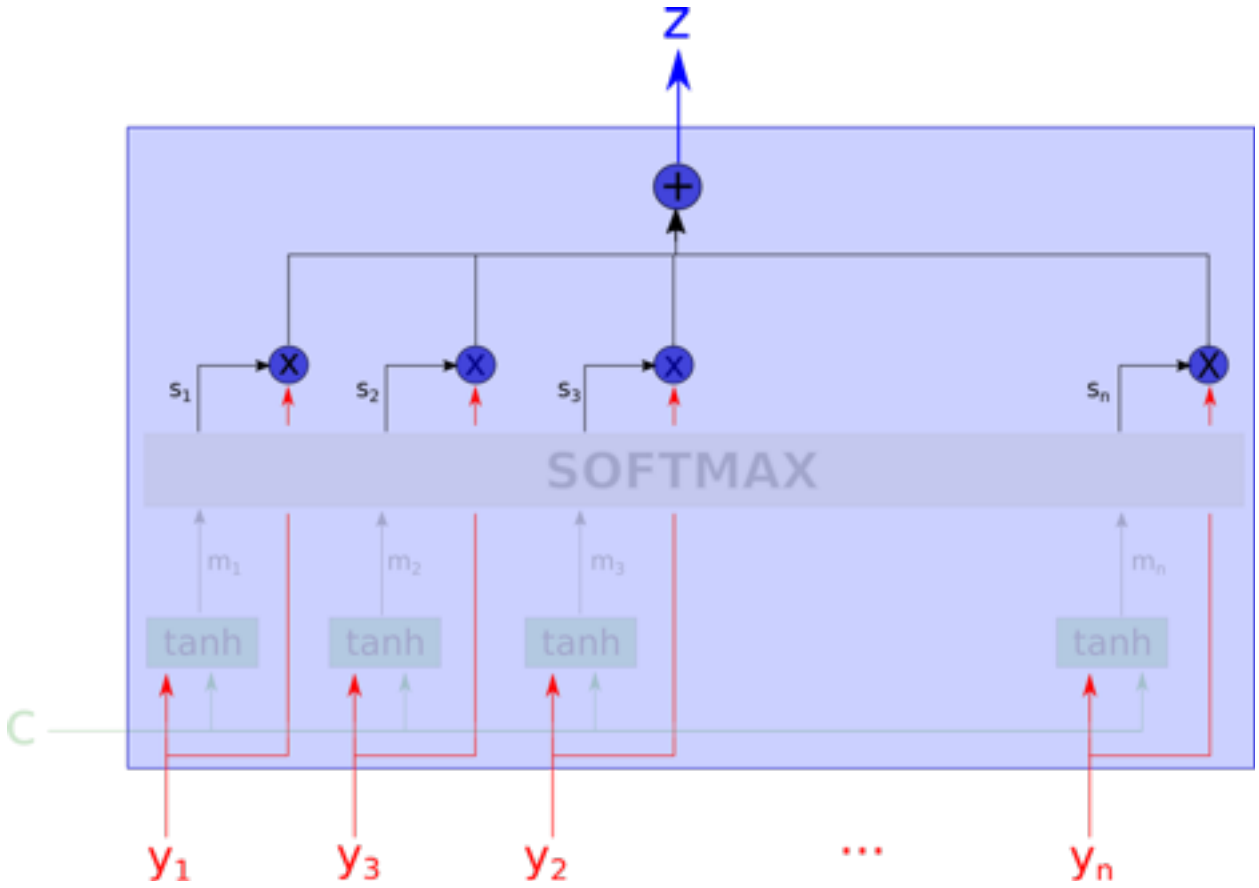


$$s_i \propto \exp\left(\langle w_m, m_i \rangle\right)$$

$$\sum_i s_i = 1$$

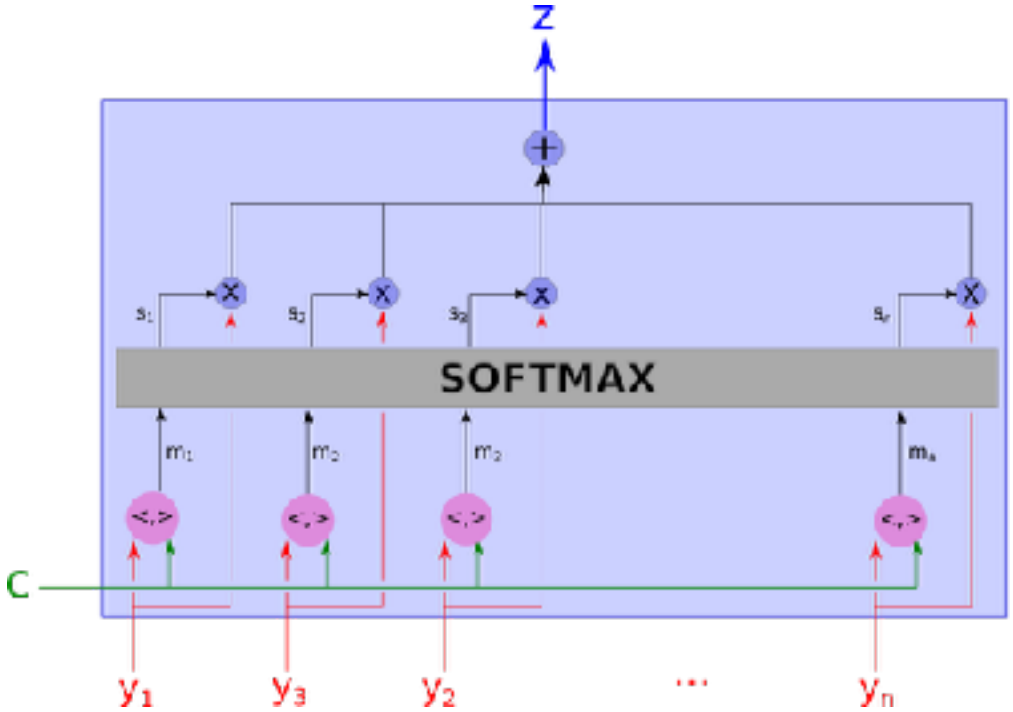
Here, the s_i are the softmax of the m_i projected on a learned direction. So the softmax can be thought as the max of the « relevance » of the variables, according to the context.

The output z is the weighted arithmetic mean of all the y_i, where the weight represent the relevance for each variable according the context c.



An other computation of « relevance »

The model presented above of an attentive model can be modified. First, the $\tanh$ layer can be replaced by any other network. The only important thing is that this function mixes up c and y_i. A version used is to compute only a dot product between $c$and y_i.



Attention Model with an other computation method for relevance.

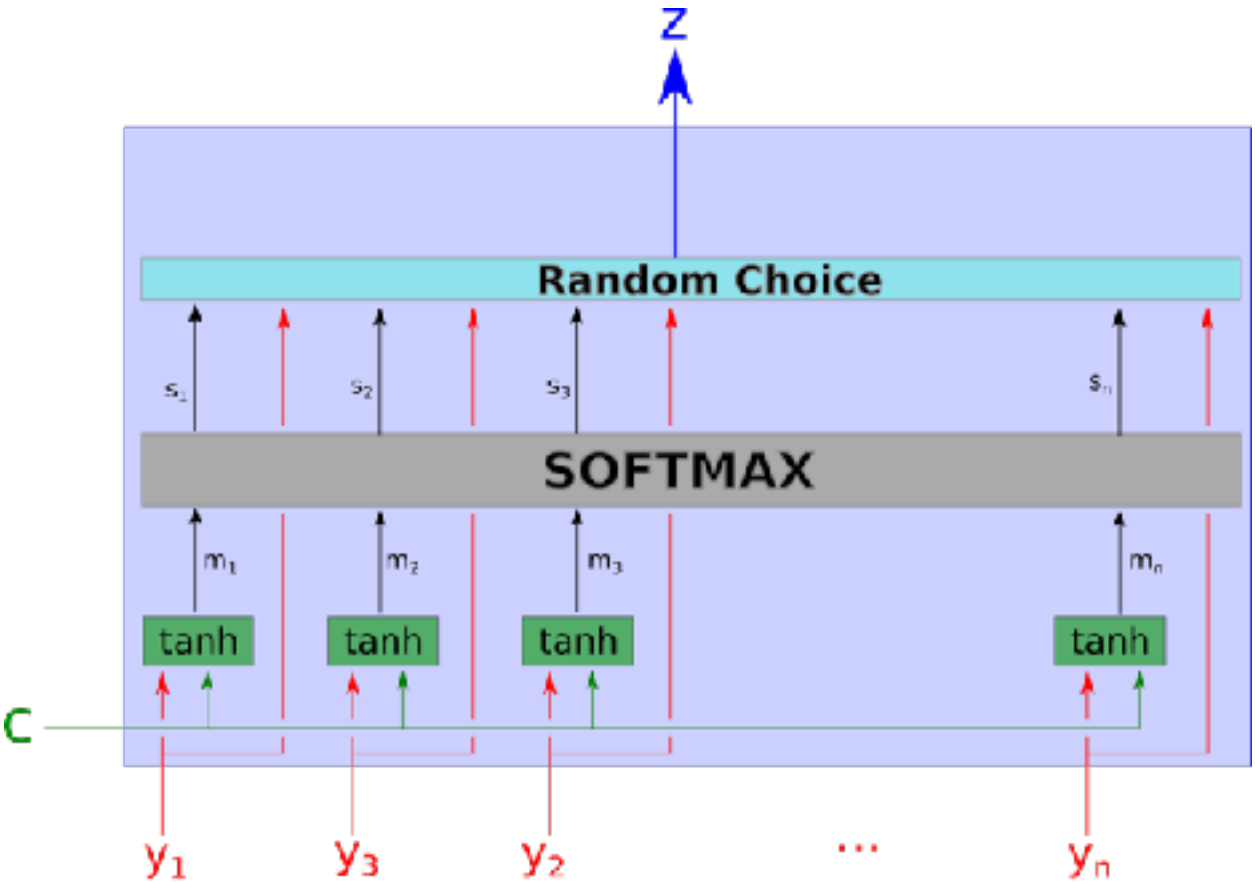
This version is even easier to understand. The attention model is « softly-choosing » the variable the most correlated with the context. As far as we know, both systems seem to produce comparable results.

An other important modification is hard attention.

Soft Attention and Hard Attention

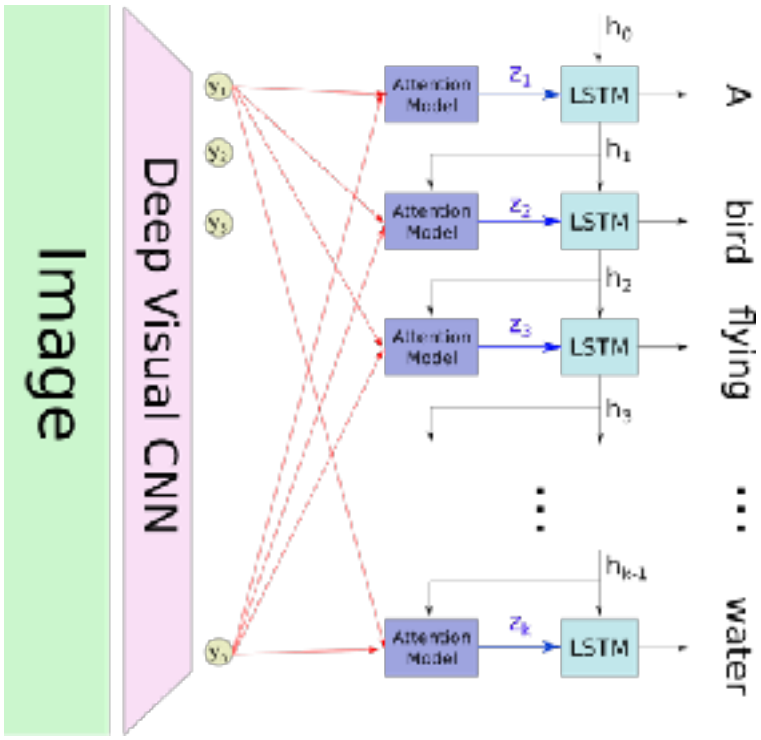
The mechanism we described previously is called « Soft attention » because it is a fully differentiable deterministic mechanism that can be plugged into an existing system, and the gradients are propagated through the attention mechanism at the same time they are propagated through the rest of the network.

Hard attention is a stochastic process: instead of using all the hidden states as an input for the decoding, the system samples a hidden state y_i with the probabilities s_i. In order to propagate a gradient through this process, we estimate the gradient by Monte Carlo sampling.



Return to the image captioning

Now, we are able to understand how the image captioning system presented before is working.



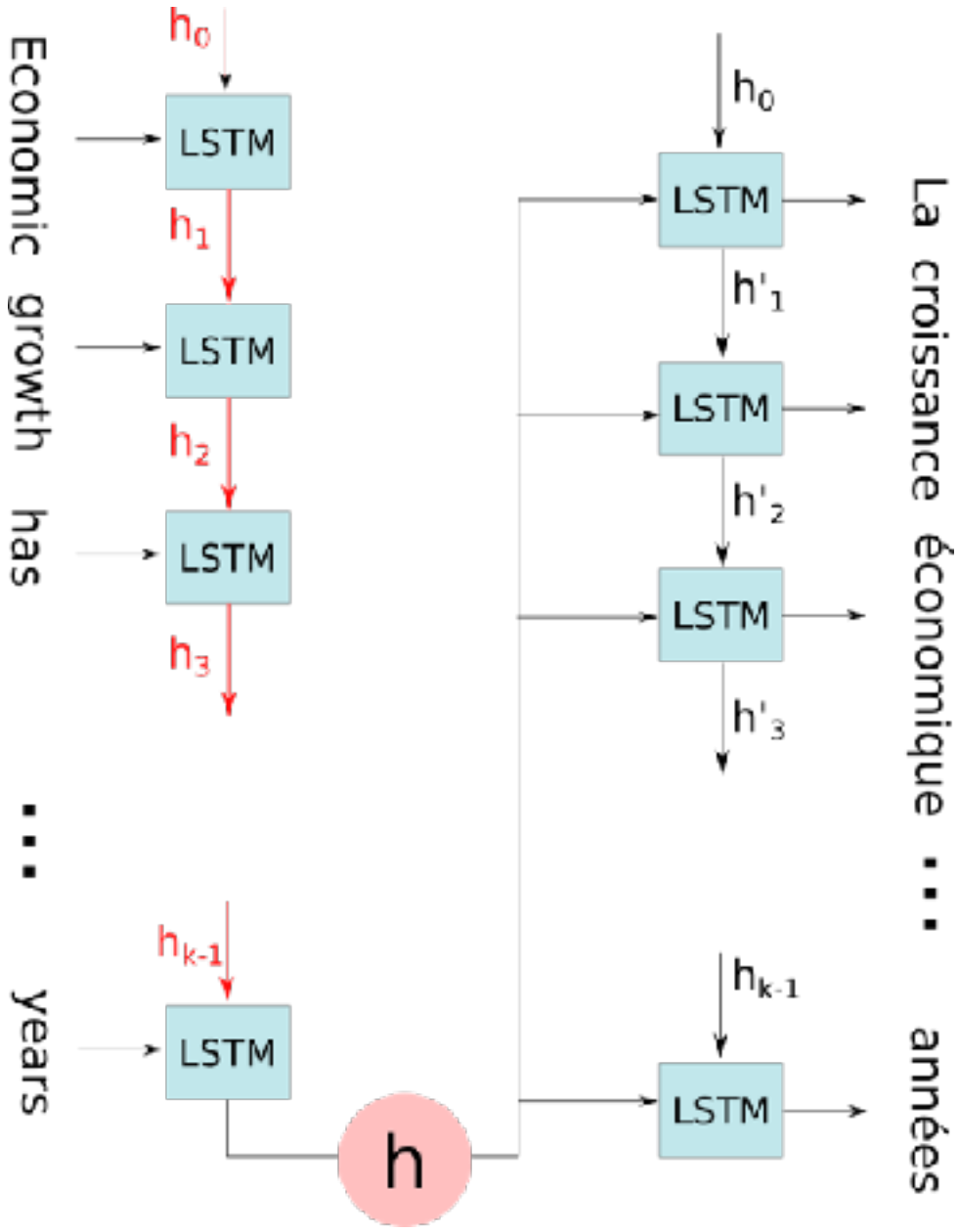
We can recognise the figure of the « classic » model for image captioning, but with a new layer of attention model. What is happening when we want to predict the new word of the caption ? If we have predicted i words, the hidden state of the LSTM is h_i. We select the « relevant » part of the image by using h_i as the context. Then, the

output of the attention model $\hat{z}_i$, which is the representation of the image filtered such that only the relevant parts of the image remains, is used as an input for the LSTM. Then, the LSTM predict a new word, and returns a new hidden state h_{i+1}.

Learning to Align in Machine Translation

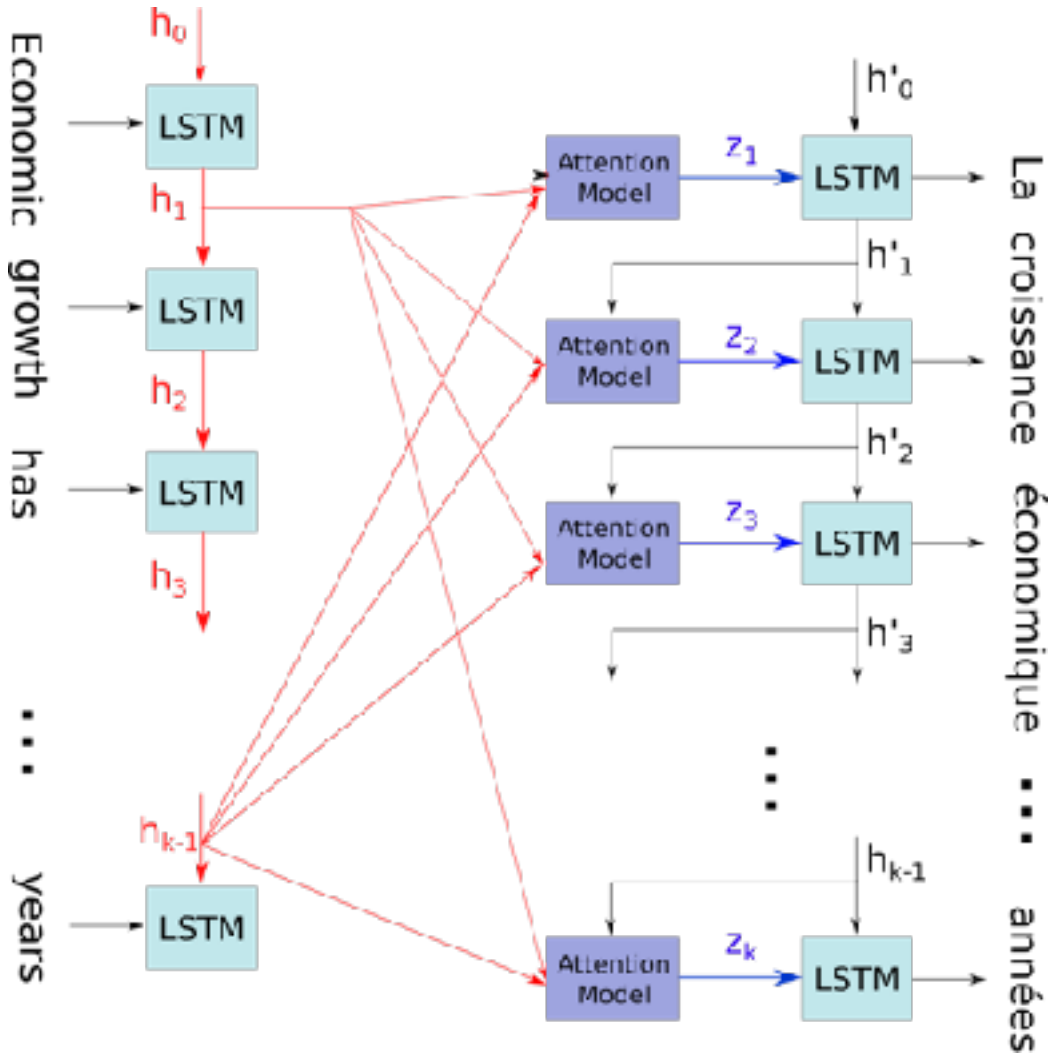
The work by Bahdanau, et al [5] proposed a neural translation model which learns to translate sentences from one language to another and introduces an attention mechanism.

Before explaining the attention mechanism, the vanilla neural translation model using an encoder-decoder works. The encoder is fed a sentence in English using Recurrent Neural Networks (RNN, usually GRU or LSTM) and produces a hidden state h. This hidden state h conditions the decoder RNN to produce the right output sentence in French.



For translation, we have the same intuition than for image captioning. When we are generating a new word, we are usually translating a single word of the original language. An attention model allows, for each new word, to focus on a part of the original text.

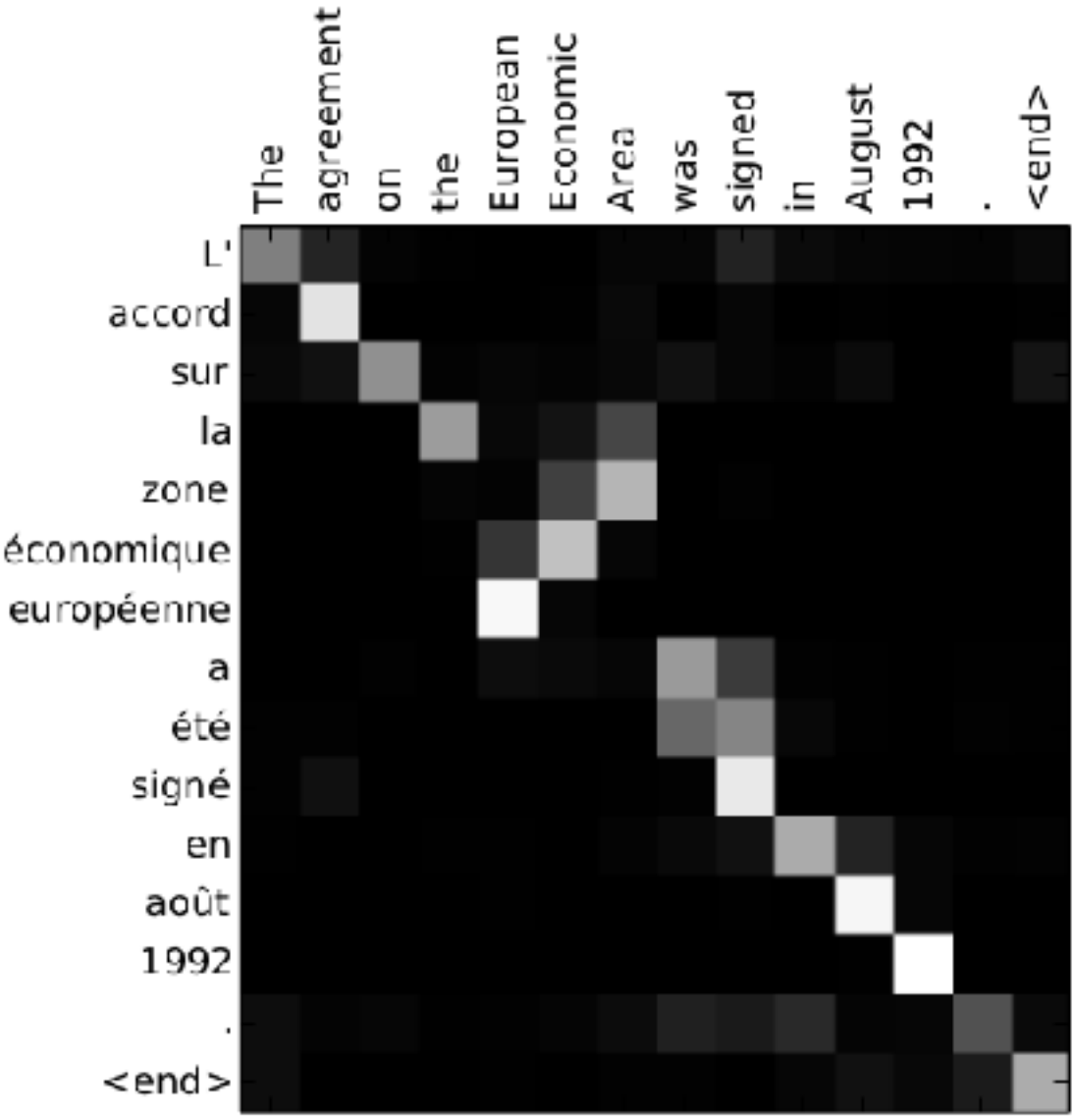
The only difference between this model and the model of image captioning is that the h_i are the successive hidden layers of a RNN.



Instead of producing just a single hidden state corresponding to the whole sentence, the encoder produces T h_j hidden states each corresponding to a word. Each time the decoder RNN produces a word, it determines the contribution of each hidden states to take as input, usually a single one (see figure below). The contribution computed using a softmax: this means that attention weights a_j are computed such that $suma_j = 1$, and all hidden states h_j contribute to the decoding with weight a_j.

In our case, the attention mechanism is fully differentiable, and does not require any additional supervision, it is simply added on top of an existing Encoder-Decoder.

This process can be seen as an alignment, because the network usually learns to focus on a single input word each time it produces an output word. This means that most of the attention weights are 0 (black) while a single one is activated (white). The image below shows the attention weights during the translation process, which reveals the alignment and makes it possible to interpret what the network has learnt (and this is usually a problem with RNNs!)



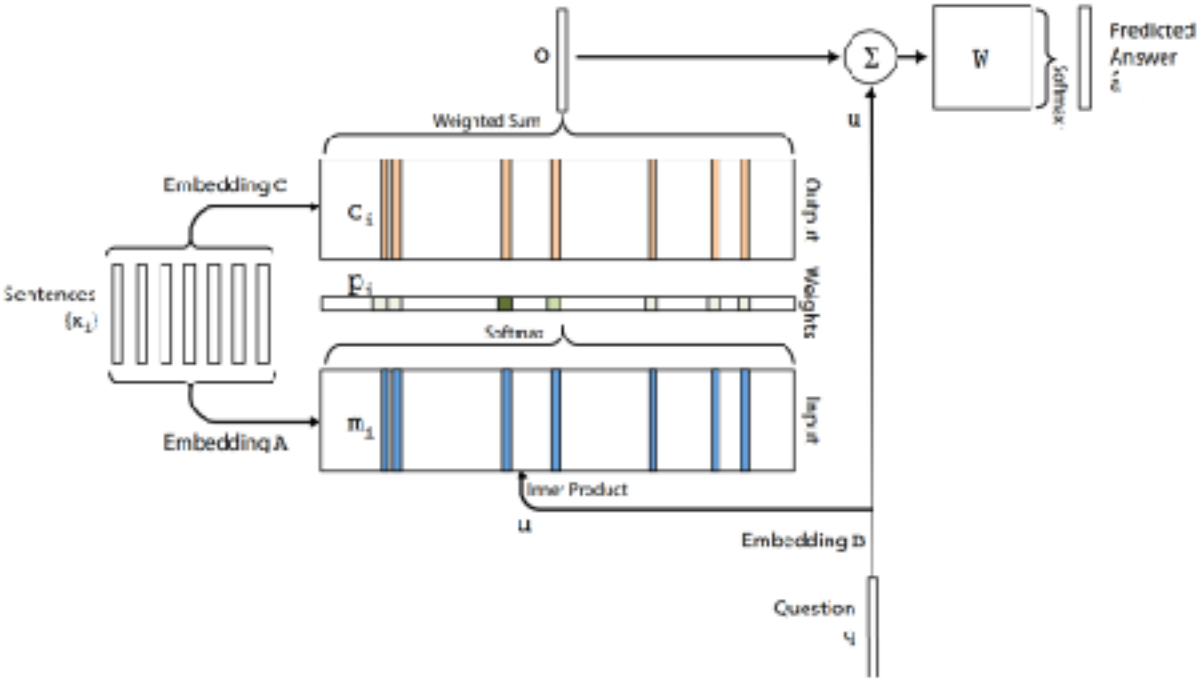
Attention without Recurrent Neural Networks

Up to now, we only described attention models in an encoder-decoder framework (i.e. with RNNs). However, when the order of input does not matter, it is possible to consider independant hidden states h_j. This is the case for instance in Raffel et Al [10], where the attention model is fully feed-forward. The same applies to the simple case of Memory Networks [6] (see next section).

From Attention to Memory Addressing

NIPS 2015 hosted a very interesting (and packed!) workshop called RAM for Reasoning, Attention and Memory. It included works on attention, but also the Memory Networks [6], Neural Turing Machines [7] or Differentiable Stack RNNs [8] and many others. These models all have in common that they use a form of external memory in which they can read (eventually write).

Comparing and explaining these models is out of the scope of this post, but the the link between attention mechanism and memory is interesting. In Memory Networks for instance, we consider an external memory – a set of facts or sentences x_i – and an input q. The network learns to address the memory, this means to select which fact x_i to focus on to produce the answer. This corresponds exactly to an attention mechanism over the external memory. In Memory Networks, the only difference is that the soft selection of the facts (blue Embedding A in the image below) is decorrelated from the weighted sum of the embeddings of the facts (pink embedding C in the image). In Neural Turing Machine, and many very recent memory based QA models, a soft attention mechanism is used. These models will be the object of a following post.



Final Word

Attention mechanism and other fully differentiable addressable memory systems are extensively studied by many researchers right now. Even though they are still young and not implemented in real world systems, they showed that they can be used to beat the state-of-the-art in many problems where the encoder-decoder framework detained the previous record.

At Heuritech, we became interested in attention mechanism a few month ago and organised a workshop to get our hands dirty and code encoder-decoder with attention mechanism. While we do not use attention mechanism in production yet, we envision it to have an important role in advanced text understanding where some reasoning is necessary, in a similar manner as the recent work by Hermann et al [9].

In a separate blog post, I will elaborate on what we've learnt during the workshop and the recent advances that were presented at the RAM workshop.

Léonard Blier et Charles Ollion