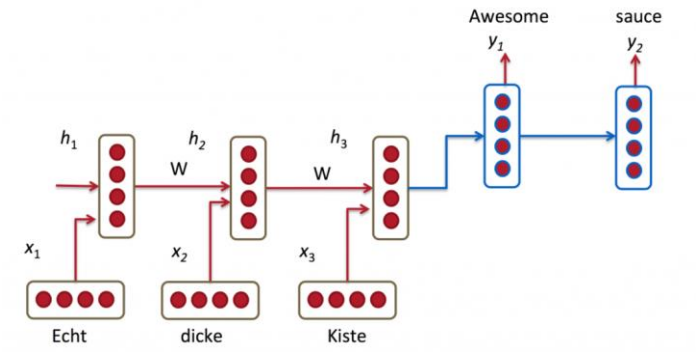
Seq2Seq Model

1. Sequence to Sequence Learning with Neural Networks, (Ilya Sutskever, Oriol Vinyals, Quoc V. Le, NIPS 2014)
2. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation(Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, Yoshua Bengio, EMNLP 2014)

Attention Mechanism

1. Neural Machin Translation By Jointly Learning to Align and Translate (Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, ICLR 2015)
2. Effective approaches to Attention-based Neural Machine Translation (Minh-Thang Luong, Hieu Pahm, Christopher D. Manning, EMNLP 2015)

Most NMT systems work by *encoding* the source sentence (e.g. a German sentence) into a vector using a Recurrent Neural Network, and then *decoding* an English sentence based on that vector, also using a RNN.

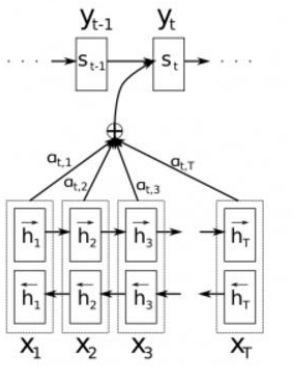


In the picture above, “Echt”, “Dicke” and “Kiste” words are fed into an encoder, and after a special signal (such as <EOS>) the decoder starts producing a translated sentence. The decoder keeps generating words until a special end of sentence token is produced.

The problem Attention solves

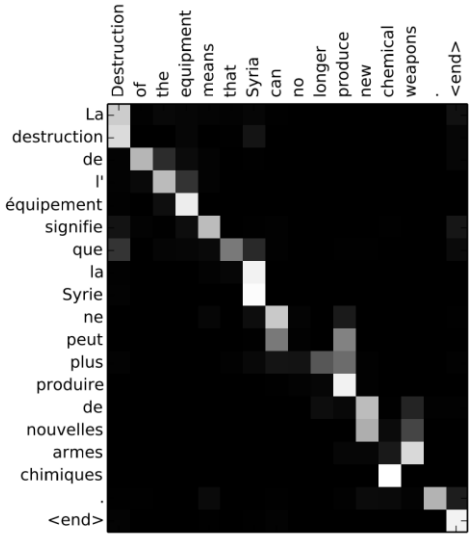
RNNs suffer from the vanishing gradient problem that prevents them from learning long-range dependencies. LSTMs improved upon this by using a **gating mechanism** that allows for explicit memory deletes and updates.

Researchers have found that reversing the source sequence (feeding it backwards into the encoder) produces significantly better results because it shortens the path from the decoder to the relevant parts of the encoder. Similarly, feeding an input sequence twice also seems to help a network to better memorize things.



With **an attention mechanism** we no longer try encode the full source sentence into a fixed-length vector. Rather, we allow the decoder to “attend” to different parts of the source sentence at each step of the output generation. Importantly, we let the model **learn** what to attend to based on the input sentence and what it has produced so far.

Visualize the model



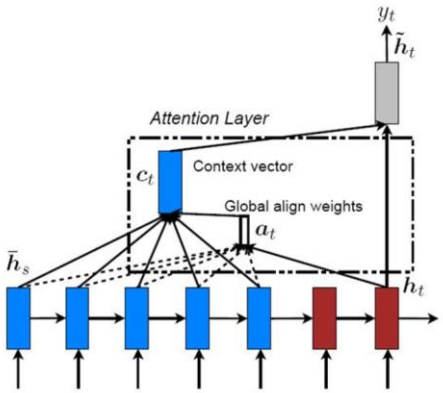
A big advantage of attention is that it gives us the ability to interpret and visualize what the model is doing. For example, by visualizing the attention weight matrix a when a sentence is translated, we can understand how the model is translating.

The Cost of Attention

We need to calculate an attention value for each combination of input and output word. Intuitively that’s equivalent outputting a translated word, and then going back through *all* of your internal memory of the text in order to decide which word to produce next.

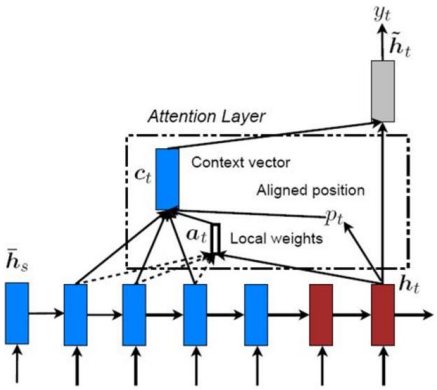
An alternative approach to attention is to use Reinforcement Learning to predict an approximate location to focus to.

It has been discussed in related paper about visual attention.



Global attention = Soft Attention Model

Step: predict each position at time t



Local attention = Soft Attention Model+ Hard Attention Model

Step1: predict an aligned position at time t

Step2: derive context vector c within the window [pt -D, pt +D]