**Paper Link**

<https://arxiv.org/pdf/1409.0473.pdf>

**Video Summary**

<https://www.youtube.com/watch?v=SysgYptB198>

𝝰1, 2 - how much attention from the encoder's second output should be paid attention to decoder at first word

Attention weights help define the context

Attention weights depend on activations from previous timestep and decoder output

**Pre-requisite papers (Roots)**

[Deep recurrent models](https://docs.google.com/document/u/1/d/1sPZQ8wtE_RbSOcb23FBHaUJM3GkBbd7eAA6SjMsS83A/edit?usp=drive_web&ouid=114045835381078446985)

Encoder-decoder architecture

Seq2Seq decoder uses a fixed vector generated by the encoder to predict the outputs, whereas bahdanau attention assigns weights between the decoder output and various vectors generated at each encoder timestep

RNN is used such that ht = f(xt , ht-1) and c = q({h1 … ht})

These represent arbitrary/generalized functions since LSTMs and SimpleRNNs calculate ht and context differently

Decoder predicts next product given context c and previously predicted words

With an RNN, each conditional probability is modeled as where g is a nonlinear multi-layered function outputting the probability of yt and st is the hidden state

Bi-directional architectures with feedforward connections

**Notes**

An annotation hj can be thought of as a representation of each input word (after embedding, of course), using a bidirectional RNN to grasp info from behind and forward

Bidirectional RNN is the encoder and the decoder emulates searching through a source sentence



si is a hidden state at time i computed as



Context vector c depends on a sequence of annotations {h1 … ht} with each annotation hi containing a strong focus on the i-th input word

Context vector is computed as a weighted sum of the annotations







eij is an alignment model scoring how well the inputs around position j and output at position i match based on the hidden state from the previous timestep and the j-th annotation hj

Notice how si-1 represents the output at position i and hj represents the input at pos j

The alignment model a is parametrized as a feedforward neural network which is jointly trained with all the other components of the system

The weighted sum over all annotations can be thought of as computing an expected annotation over possible alignments — the average "meaning" of the word is its context

𝝰ij is the probability that the target word yi is aligned to a source word xj — the i-th context vector is the expected annotation over all annotations with probability given by 𝝰ij

The probability aij or its associated energy eij reflects the importance of hj with respect to the previous hidden state st-1 in deciding si and generating yi

This makes the decoder choose which parts of the source sentence to pay attention to

Note how si and yi both depend on the context which depends on the attention weights, and then the attention weights depend on si-1 so the outputs are self-reliant to some extent

This serves as better memory since the encoder does not have to hold all the context in a single fixed length vector (the final context vector) and instead the decoder can look at the annotations and selectively retrieve

In order to make the annotation corresponding to each word summarize both the previous words and future words, we use a bidirectional RNN

Concatenate the forward hidden states and backward hidden states, each calculated in reverse order, and use the results (annotations for each input timestep) to compute the context vector

**Branches**

Principles improved in "Attention Is All You Need"

Attention Is All You Need does not use timesteps / RNN

The alignment model a is a feed-forward network but it is not a latent variable

* How does this relate to latent variables? (see DL book pt 3)