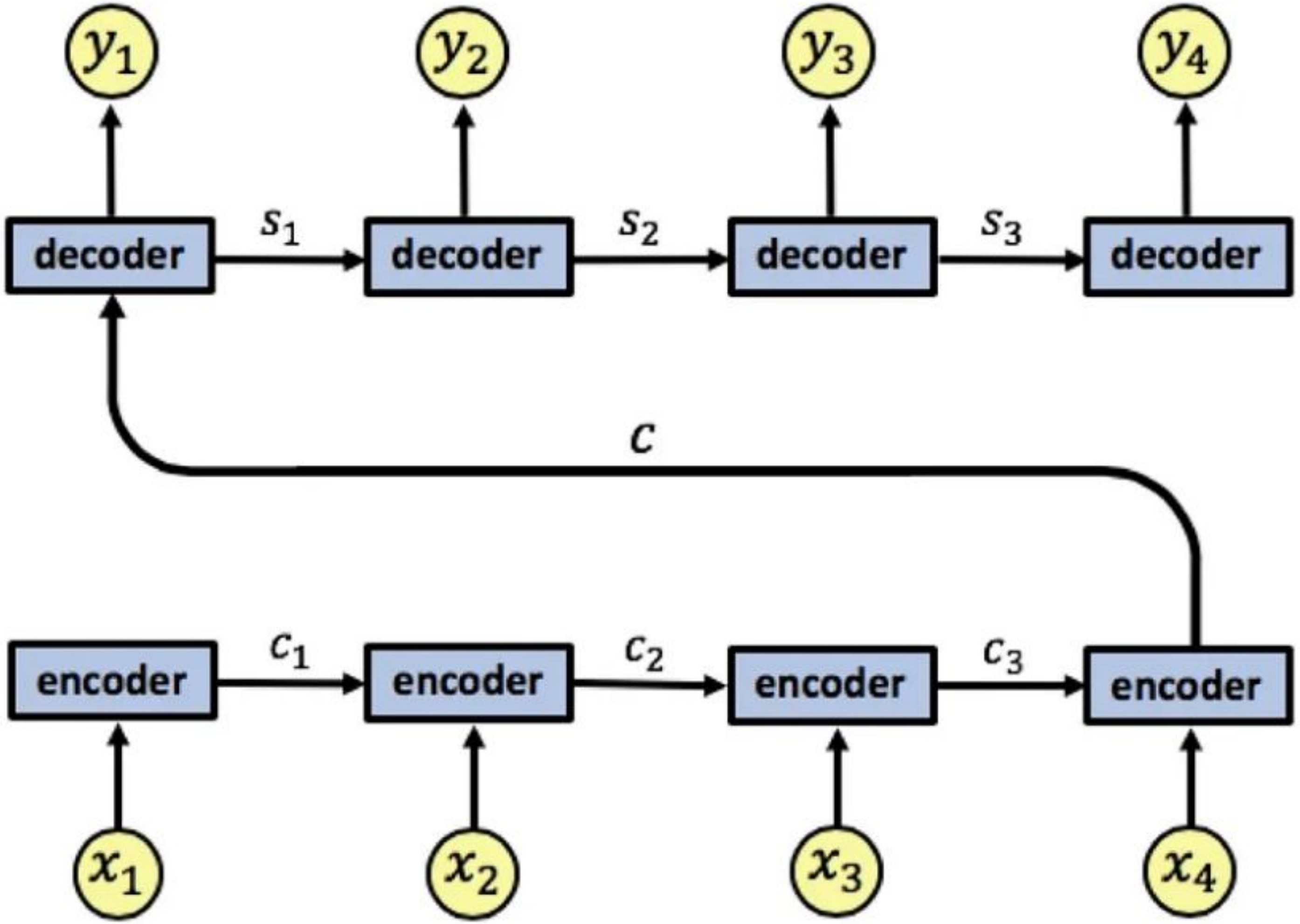
Attention Models

Arin Ghazarian
Chapman University

Attention mechanism

- In an LSTM based encoder-decoder model for English to French translator, in order to produce the first word of the French output, the decoder network is given only the state vector after processing the last English word from the last unit of encoder.
- In theory, the state vector from the last unit in encoder can encode information and context about the whole English sentence, but in practice, this information is often poorly preserved by the LSTM.
- For long and complex sequences, a single fixed-length state vector is not enough to capture and preserve all the necessary context

Regular RNN Encoder-Decoder

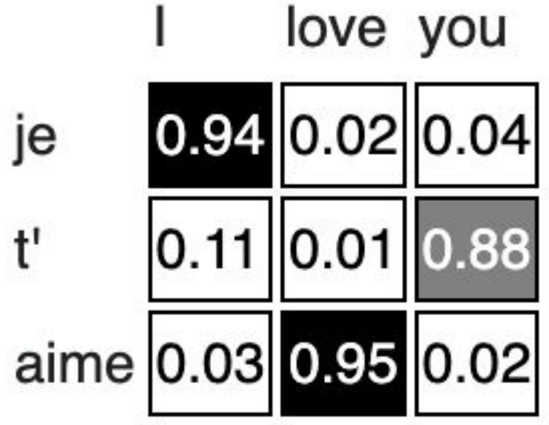


Attention Mechanism

- Bahdanau et al. (2014), "Neural Machine Translation by Jointly Learning to Align and Translate"
- Invented to improve the performance of the encoder-decoder based language translation models
- Allow language models to focus on a particular part of the observed context at each time step
- Tries to mimic cognitive attention.
- Assigns higher weights to the most relevant parts of the input (small but important) sequence in decoder
- The attention layer can access all previous states and weigh them according to a learned measure of relevance, providing relevant information about far-away tokens.
- Using an attention mechanism in th example English-French translator model, the decoder is given access to the state vectors of every token in the English sentence and not just the last state vector, and also learns the attention weights which tell how much to attend to each input state vector.

Attention Weights

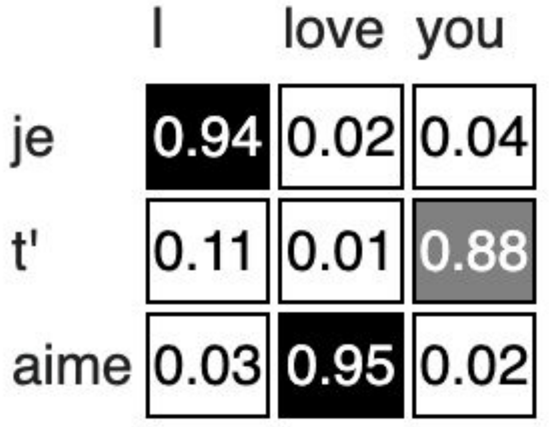
- Viewed as a matrix, the attention weights show how the network adjusts its focus according to context.



	I	love	you
je	0.94	0.02	0.04
t'	0.11	0.01	0.88
aime	0.03	0.95	0.02

Attention Weights

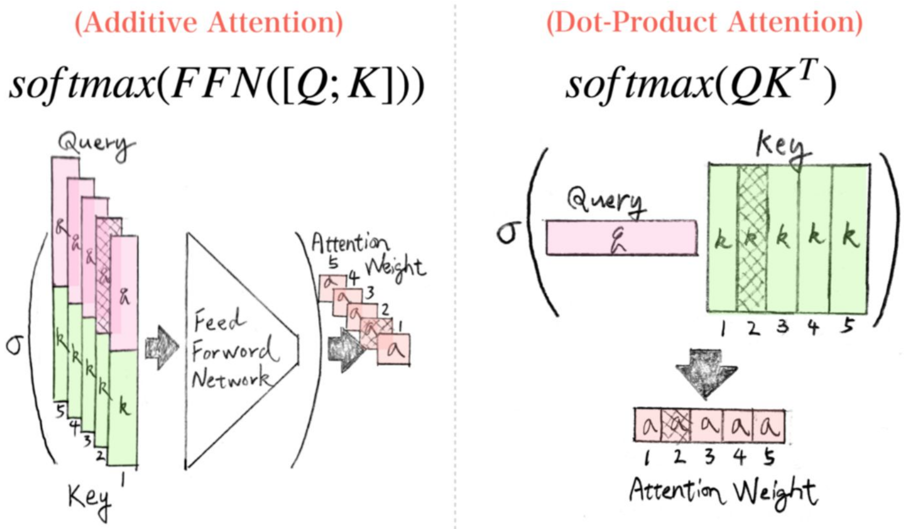
- On the first pass through the decoder, 94% of the attention weight is on the first English word "I", so the network offers the word "je". On the second pass of the decoder, 88% of the attention weight is on the third English word "you", so it offers "t'". On the last pass, 95% of the attention weight is on the second English word "love", so it offers "aime".



	I	love	you
je	0.94	0.02	0.04
t'	0.11	0.01	0.88
aime	0.03	0.95	0.02

Attention mechanism Variations

- **Dot-Product attention:** measures attention scores using dot product between the query and key vectors.
  - https://arxiv.org/abs/1508.04025
- **Additive attention mechanism (Bahdanau):** Additive Attention computes attention scores by applying a feed-forward neural network to the concatenated query and key vectors.



https://pretteyandnerdy.wordpress.com/2019/04/26/attention-is-all-you-need/

Attention mechanism

- Given a sequence of tokens t_i
  - We compute a soft weight w_i for each t_i. w_i is nonnegative and $\sum_i w_i = 1.$
  - Each t_i is assigned a value vector v_i which is computed using the word embedding of t_i
  - The weighted average of v_i is the output of the attention mechanism:

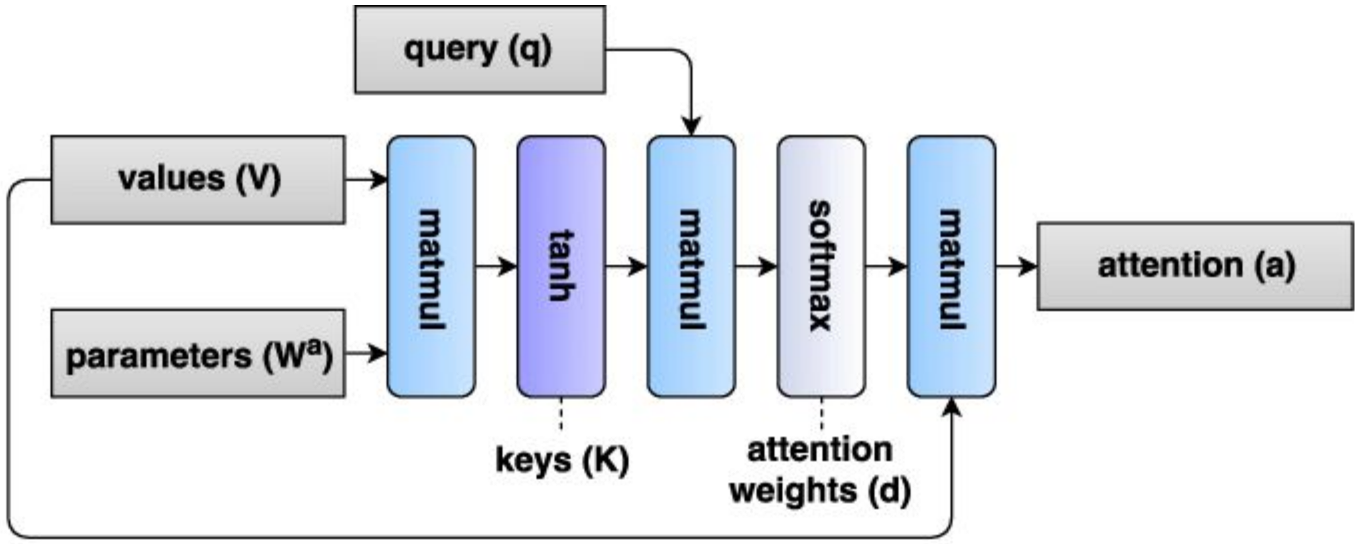
$$\sum_i w_i v_i$$

Attention mechanism

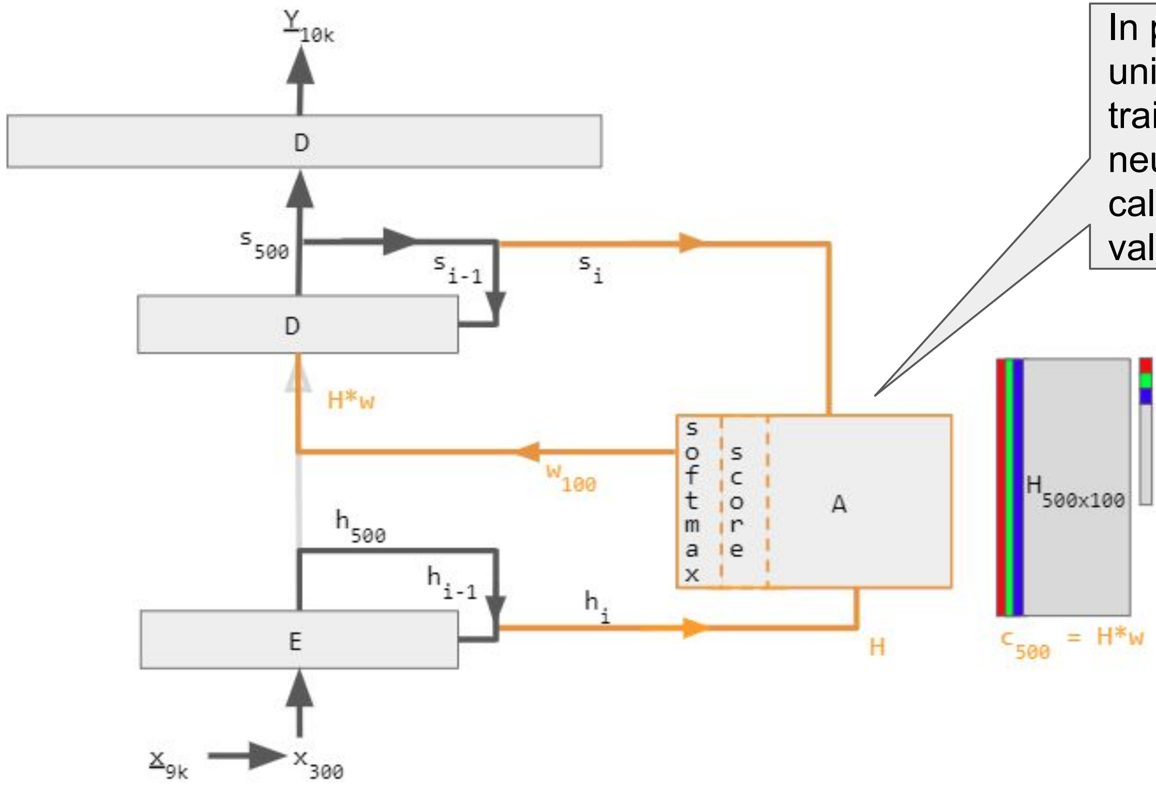
- The query-key mechanism computes the soft weights (w_i)
  - The weights are not hard and pre-calculated, they change with the input
- From the word embedding of each token, it computes its corresponding query vector q_i and key vector k_i.
- The weights are calculated by applying the softmax function to the dot product $q_i.k_j$ where i represents the current token and j represents the token that's being attended to:

$$w_i=\text{softmax}(q_i.k_j)$$

Additive Attention



source

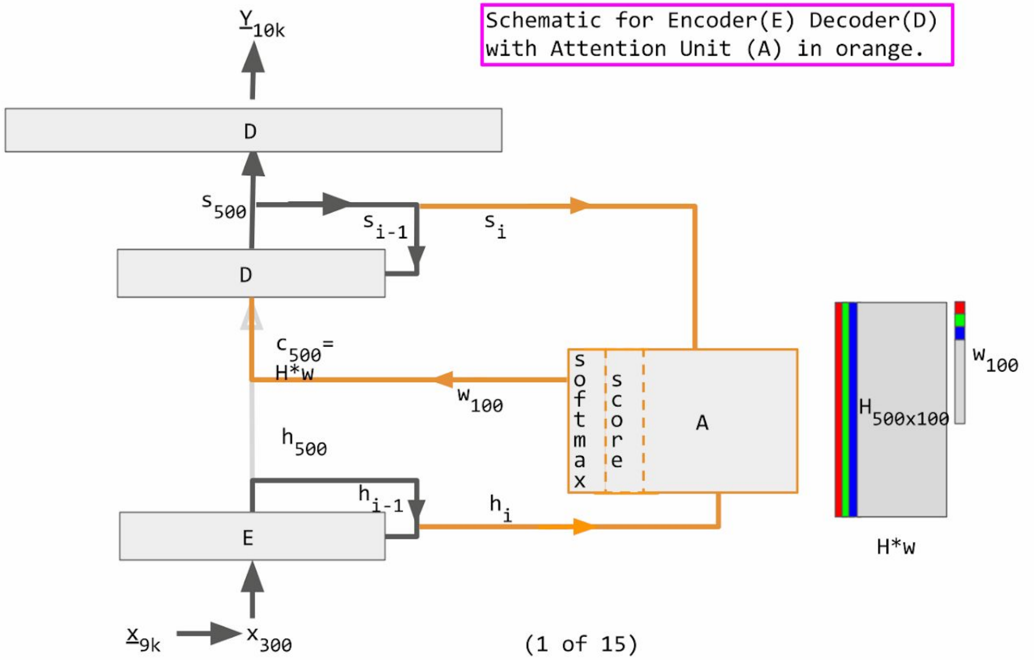


In practice, the attention unit consists of 3 trained, fully-connected neural network layers called query, key, and value.

Encoder-decoder with attention. The left part (black lines) is the encoder-decoder, the middle part (orange lines) is the attention unit, and the right part (in grey & colors) is the computed data. Grey regions in H matrix and w vector are zero values. Numerical subscripts indicate vector sizes while lettered subscripts i and i − 1 indicate time steps.

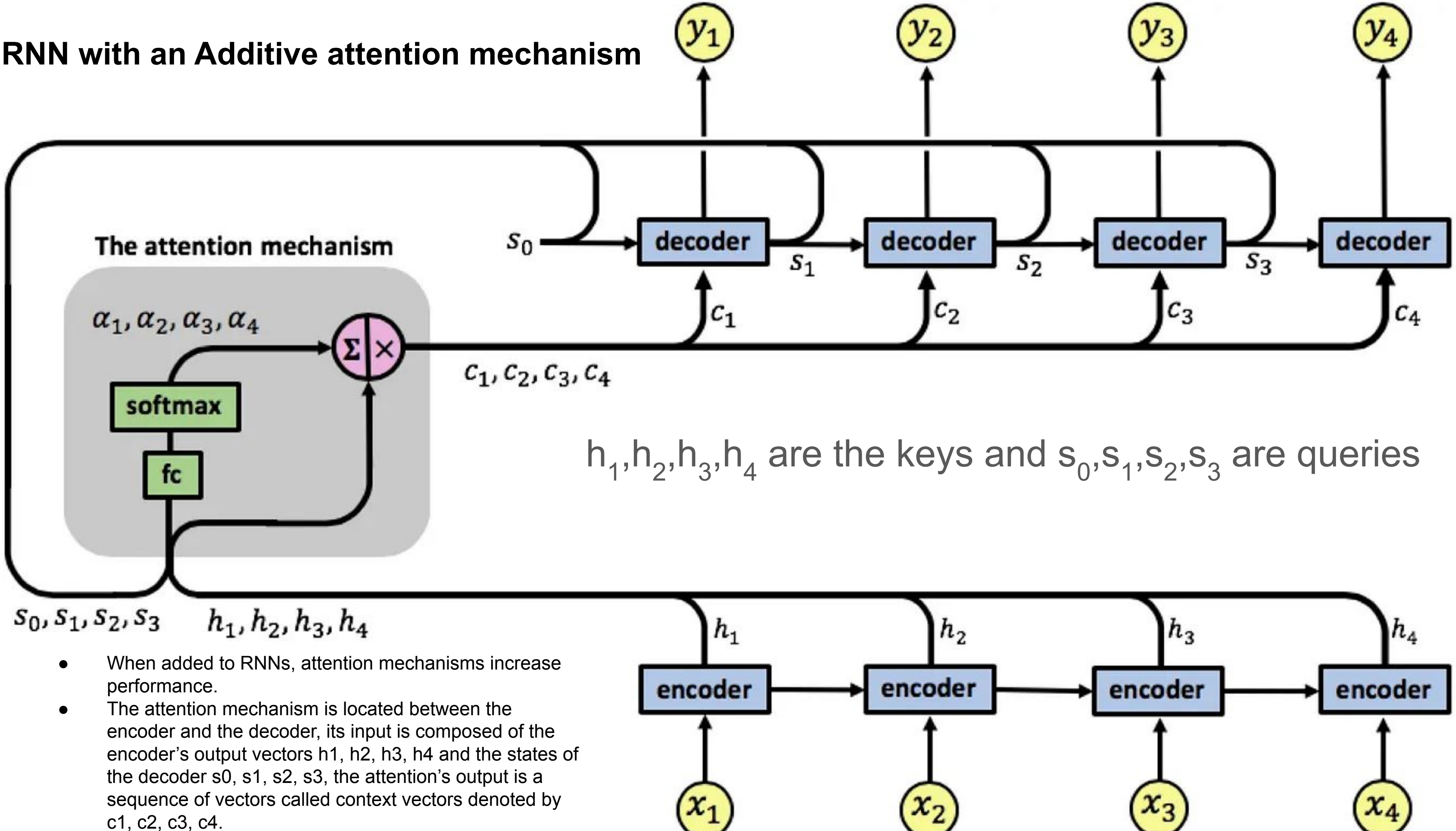
Source: https://en.wikipedia.org/wiki/Attention_(machine_learning)

Label	Description
100	Max. sentence length
300	Embedding size (word dimension)
500	Length of hidden vector
9k, 10k	Dictionary size of input & output languages respectively.
x, Y	9k and 10k 1-hot dictionary vectors. x → x implemented as a lookup table rather than vector multiplication. Y is the 1-hot maximizer of the linear Decoder layer D; that is, it takes the argmax of D's linear layer output.
x	300-long word embedding vector. The vectors are usually pre-calculated from other projects such as GloVe or Word2Vec.
h	500-long encoder hidden vector. At each point in time, this vector summarizes all the preceding words before it. The final h can be viewed as a "sentence" vector, or a thought vector as Hinton calls it.
s	500-long decoder hidden state vector.
E	500 neuron RNN encoder. 500 outputs. Input count is 800–300 from source embedding + 500 from recurrent connections. The encoder feeds directly into the decoder only to initialize it, but not thereafter; hence, that direct connection is shown very faintly.
D	2-layer decoder. The recurrent layer has 500 neurons and the fully-connected linear layer has 10k neurons (the size of the target vocabulary).[12] The linear layer alone has 5 million (500 × 10k) weights -- ~10 times more weights than the recurrent layer.
score	100-long alignment score
w	100-long vector attention weight. These are "soft" weights which changes during the forward pass, in contrast to "hard" neuronal weights that change during the learning phase.
A	Attention module — this can be a dot product of recurrent states, or the query-key-value fully-connected layers. The output is a 100-long vector w.
H	500×100. 100 hidden vectors h concatenated into a matrix
c	500-long context vector = H * w. c is a linear combination of h vectors weighted by w.



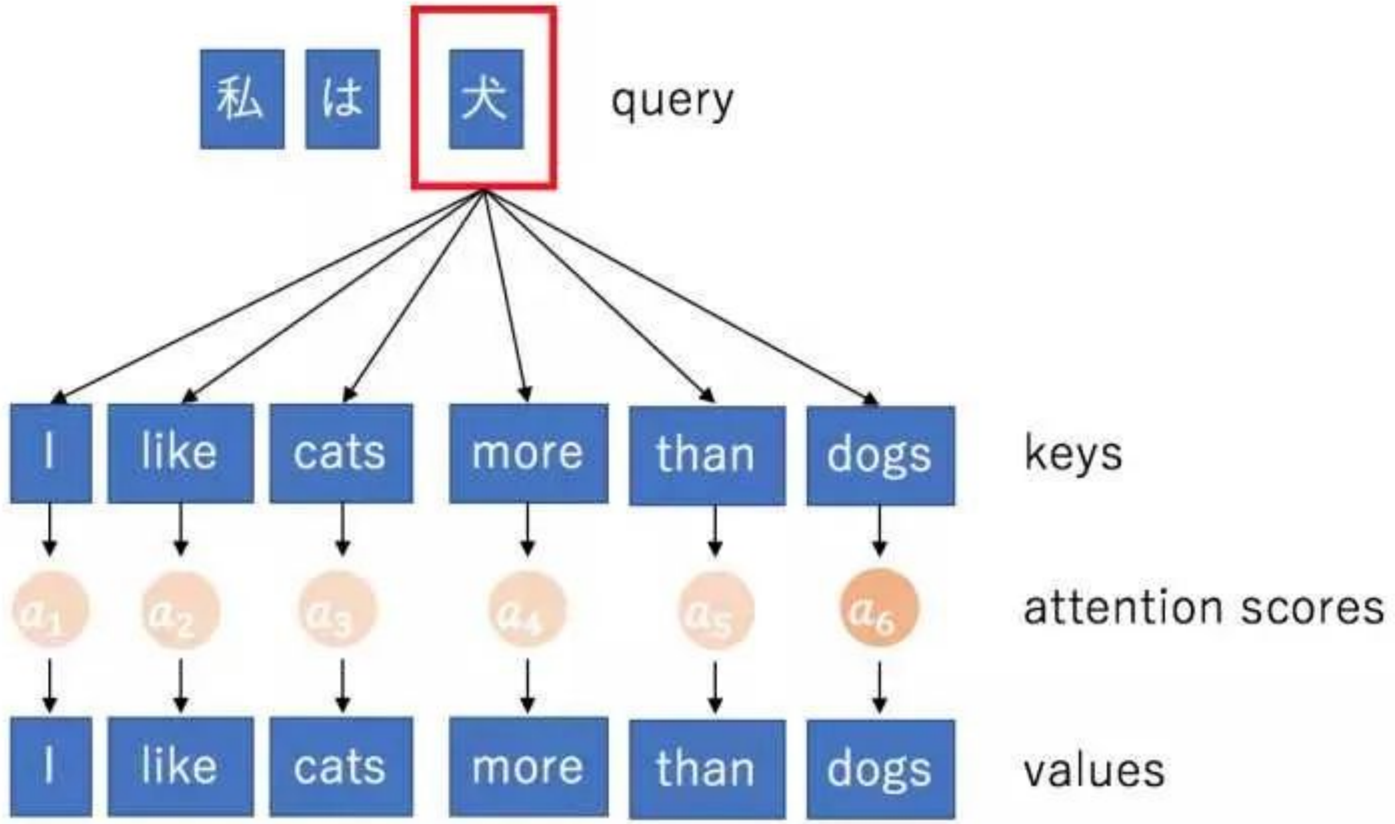
Source: https://en.wikipedia.org/wiki/Attention_(machine_learning)

RNN with an Additive attention mechanism



h_1,h_2,h_3,h_4 are the keys and s_0,s_1,s_2,s_3 are queries

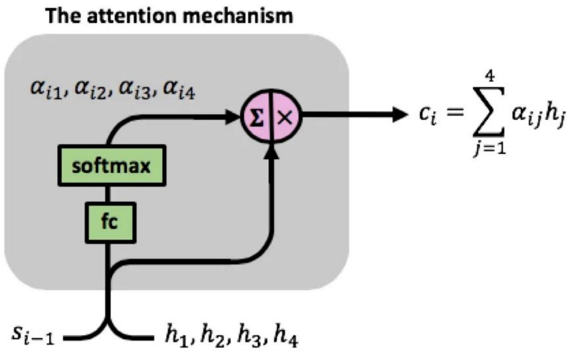
- When added to RNNs, attention mechanisms increase performance.
- The attention mechanism is located between the encoder and the decoder, its input is composed of the encoder’s output vectors h1, h2, h3, h4 and the states of the decoder s0, s1, s2, s3, the attention’s output is a sequence of vectors called context vectors denoted by c1, c2, c3, c4.


私
は
犬
query
I
like
cats
more
than
dogs
keys
a_1
a_2
a_3
a_4
a_5
a_6
attention scores
I
like
cats
more
than
dogs
values

source

RNN with an Additive attention mechanism

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


The attention mechanism
$\alpha_{i1}, \alpha_{i2}, \alpha_{i3}, \alpha_{i4}$
softmax
fc
Σ $\times$
$c_i = \sum_{j=1}^{4} \alpha_{ij} h_j$
s_{i-1}
h_1, h_2, h_3, h_4

- At time step i, the mechanism has h_1,h_2,h_3,h_4 and s_{i-1} as inputs, it uses the fc neural network and the softmax function to compute the attention weights α_{i1}, α_{i2}, α_{i3}, α_{i4}, these are then used in the computation of the context vector c_i.

General Attention Mechanism

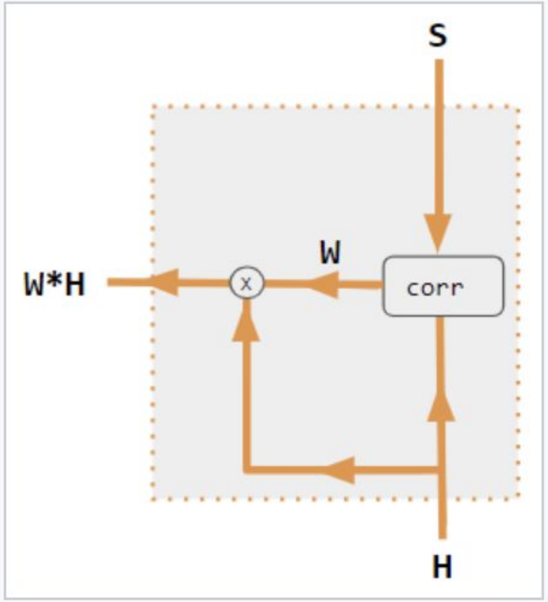
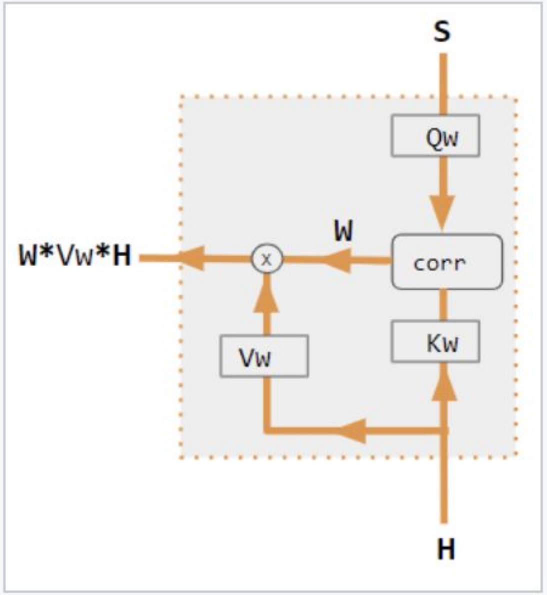
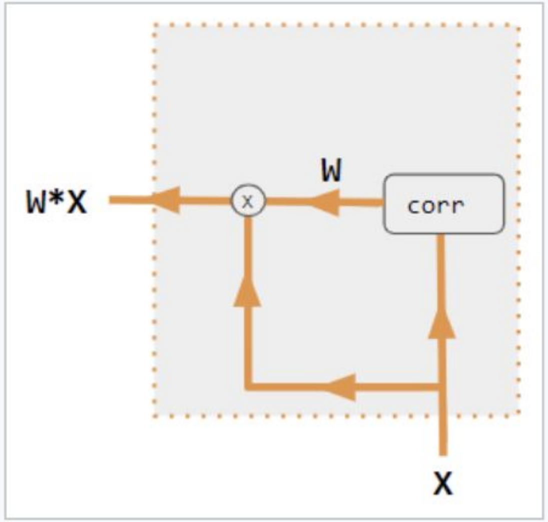
- The General Attention Mechanism The general attention mechanism makes use of three main components, namely the queries, the keys, and the values
- The general attention mechanism then performs the following computations:
  - Each query vector, $q=s_{t-1}$, is matched against a database of keys to compute a score value. This matching operation is computed as the dot product of the specific query under consideration with each key vector, k_i :

    $$e_{q,ki}=q\cdot k_i$$

  - The scores are passed through a softmax operation to generate the weights: $\alpha_{q,ki}=\text{softmax}(e_{q,ki})$
  - The generalized attention is then computed by a weighted sum of the value vectors, v_{ki} , where each value vector is paired with a corresponding key:

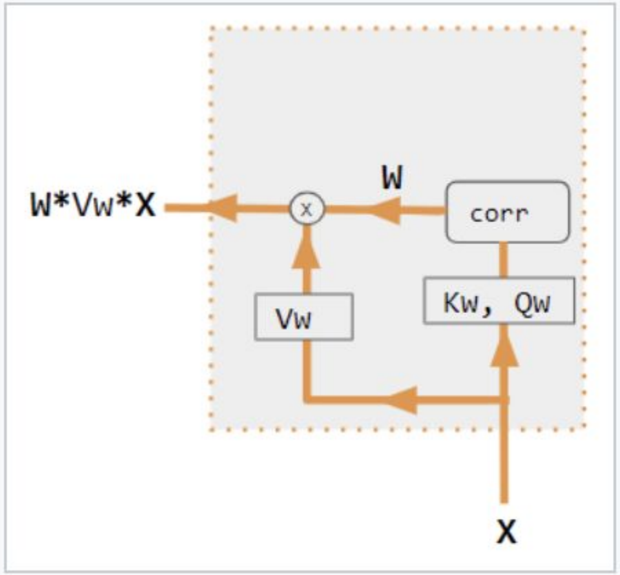
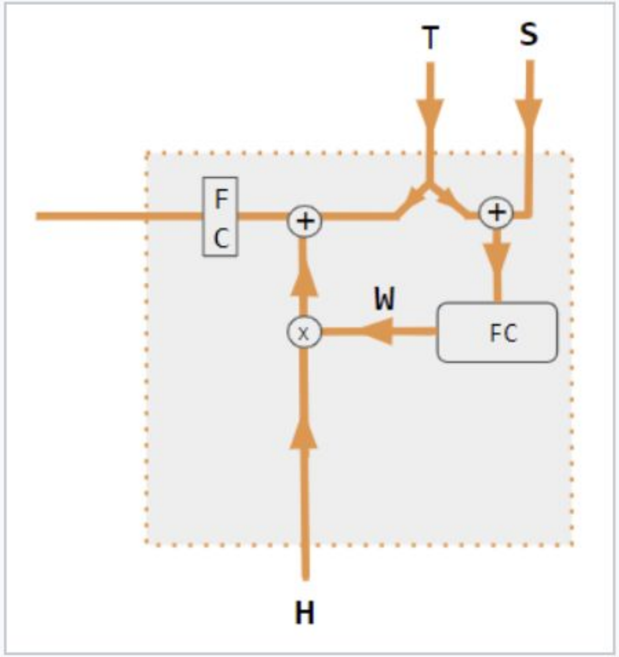
    $$\text{attention}(q,K,V)=\sum\alpha_{q,ki}v_{ki}\ .$$

Variants of Attention

1. encoder-decoder dot product	2. encoder-decoder QKV (General attention)	3. encoder-only dot product
 Both encoder & decoder are needed to calculate attention.[14]	 Both encoder & decoder are needed to calculate attention.[20]	 Decoder is *not* used to calculate attention. With only 1 input into corr, W is an auto-correlation of dot products. $w_{ij} = x_i x_j$[21]

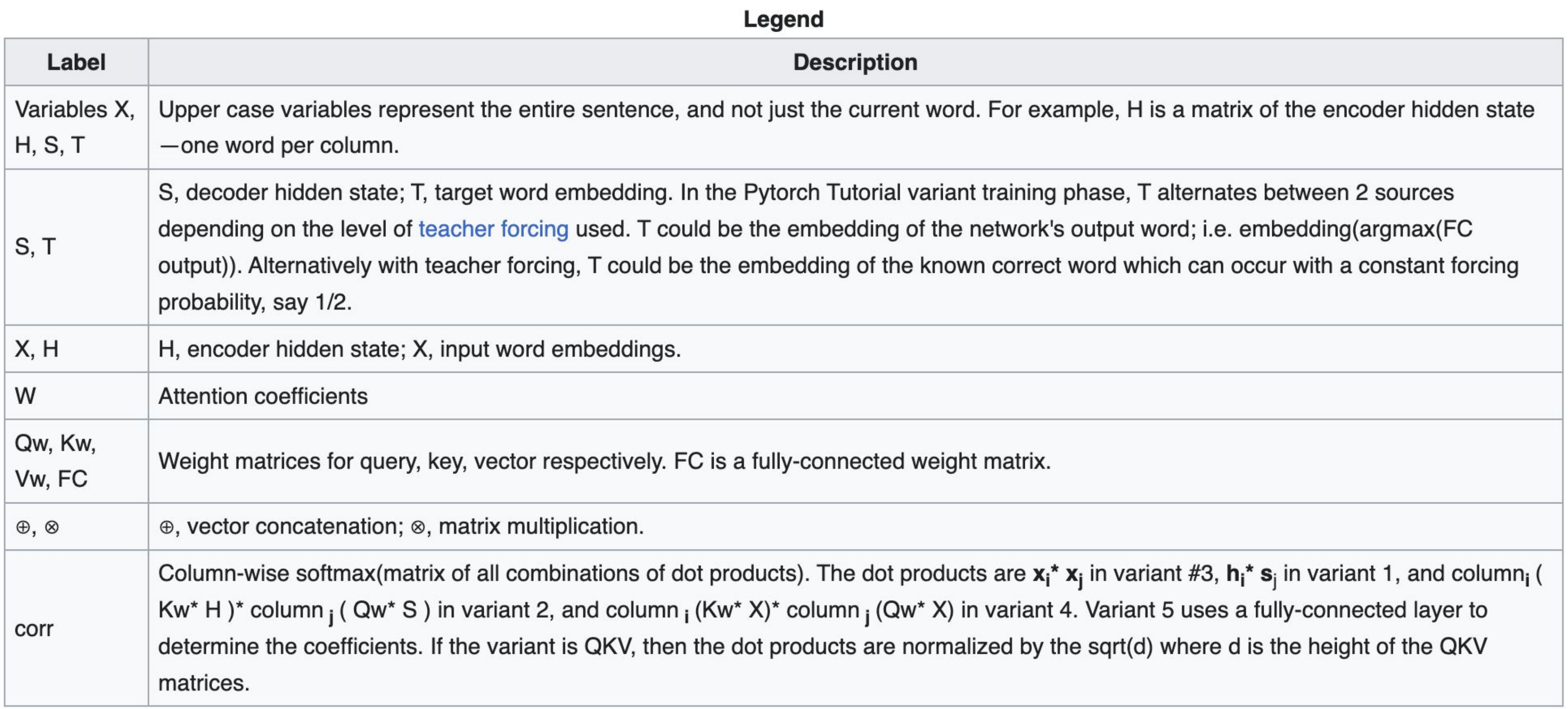
https://en.wikipedia.org/wiki/Attention_(machine_learning)

Variants of Attention

4. encoder-only QKV	5. Pytorch tutorial
 Decoder is *not* used to calculate attention.[22]	 A fully-connected layer is used to calculate attention instead of dot product correlation.[23]

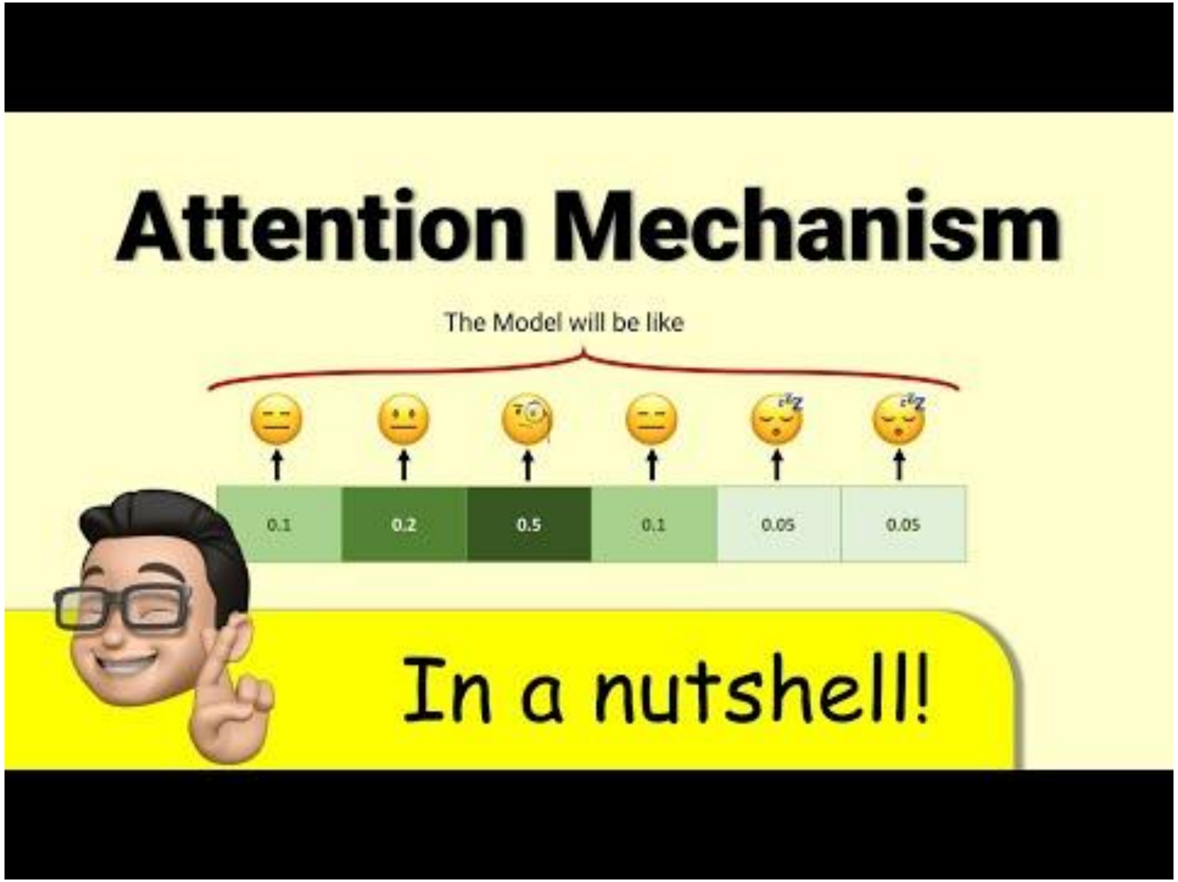
https://en.wikipedia.org/wiki/Attention_(machine_learning)

Variants of Attention

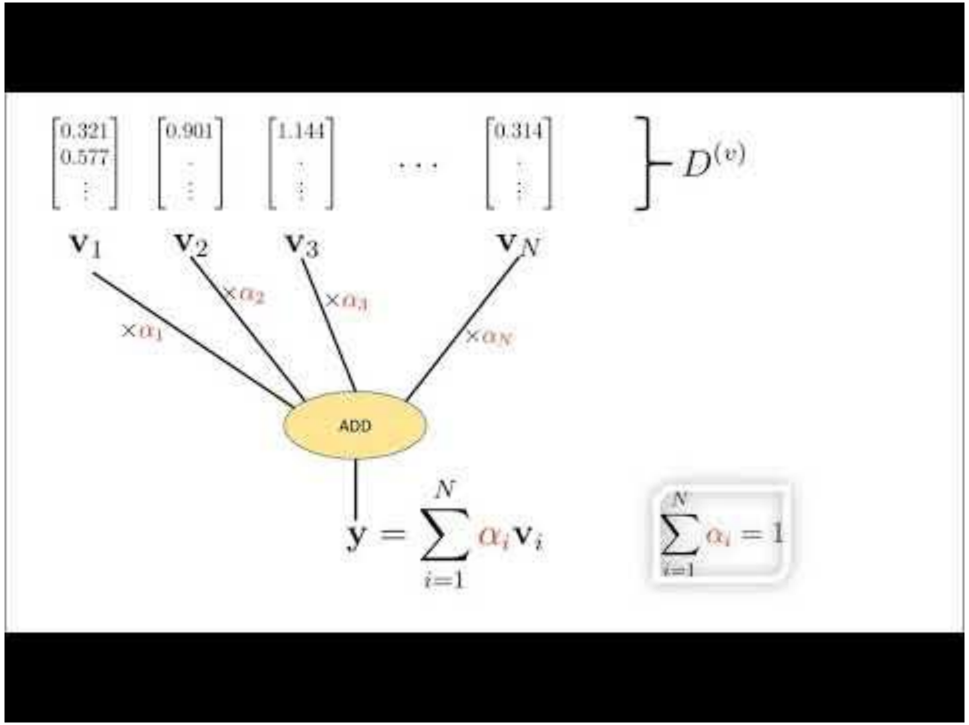


Legend

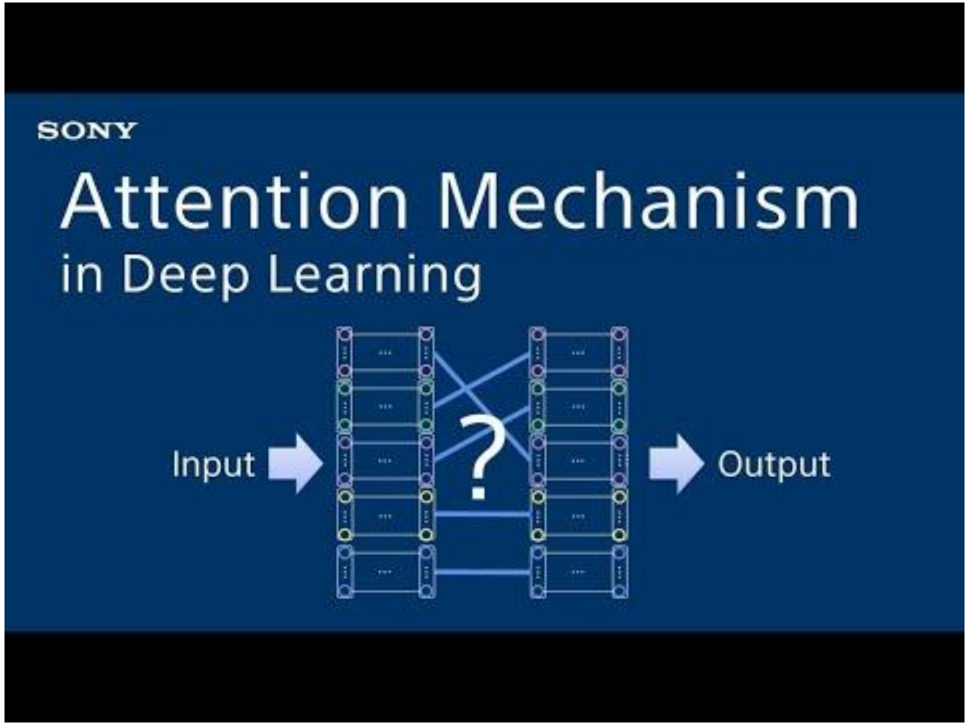
Label	Description
Variables X, H, S, T	Upper case variables represent the entire sentence, and not just the current word. For example, H is a matrix of the encoder hidden state—one word per column.
S, T	S, decoder hidden state; T, target word embedding. In the Pytorch Tutorial variant training phase, T alternates between 2 sources depending on the level of teacher forcing used. T could be the embedding of the network's output word; i.e. embedding(argmax(FC output)). Alternatively with teacher forcing, T could be the embedding of the known correct word which can occur with a constant forcing probability, say 1/2.
X, H	H, encoder hidden state; X, input word embeddings.
W	Attention coefficients
Qw, Kw, Vw, FC	Weight matrices for query, key, vector respectively. FC is a fully-connected weight matrix.
⊕, ⊗	⊕, vector concatenation; ⊗, matrix multiplication.
corr	Column-wise softmax(matrix of all combinations of dot products). The dot products are $\mathbf{x}_i$* $\mathbf{x}_j$ in variant #3, $\mathbf{h}_i$* $\mathbf{s}_j$ in variant 1, and column$_i$ (Kw* H)* column $_j$ (Qw* S) in variant 2, and column $_i$ (Kw* X)* column $_j$ (Qw* X) in variant 4. Variant 5 uses a fully-connected layer to determine the coefficients. If the variant is QKV, then the dot products are normalized by the sqrt(d) where d is the height of the QKV matrices.



https://www.youtube.com/watch?v=oMeIDqRguLY



https://www.youtube.com/watch?v=wj3ZYbKKUHI



https://www.youtube.com/watch?v=aQQa_xl78Qw

Demo:Translation with a Sequence to Sequence Network and Attention

- NLP From Scratch: Translation with a Sequence to Sequence Network and Attention

Online Resources

- https://medium.datadriveninvestor.com/attention-in-rnns-321fbcd64f05
- The General Attention Mechanism with NumPy and SciPy.
- A Comprehensive Guide to Attention Mechanism in Deep Learning for Everyone