Copyright Notice

deeplearning.ai

Sequence to sequence models

Transformers Intuition

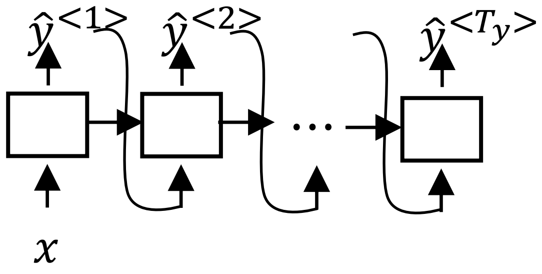
Transformers Motivation

As the complexity of your sequence task increases, so does the complexity of your model.

Increased complexity,
sequential

All these models are still sequential.

RNN

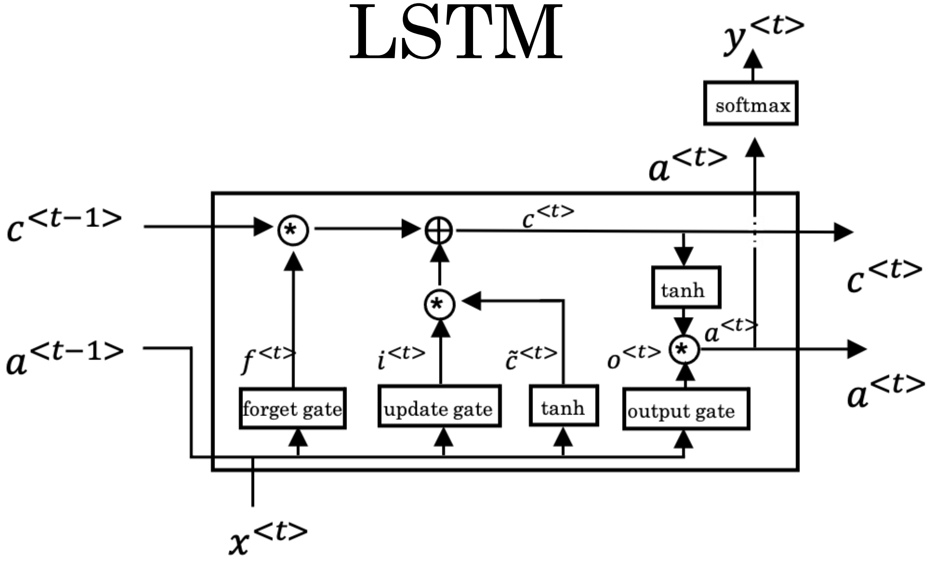


RNN has problems with vanishing gradients, which made it hard to capture long range dependencies and sequences.

GRU

As a way to resolve many of those problems, we looked at the GRU and LSTM

LSTM



Using transformer architecture, we can run a lot of computtions for an entire sequnce in parallel

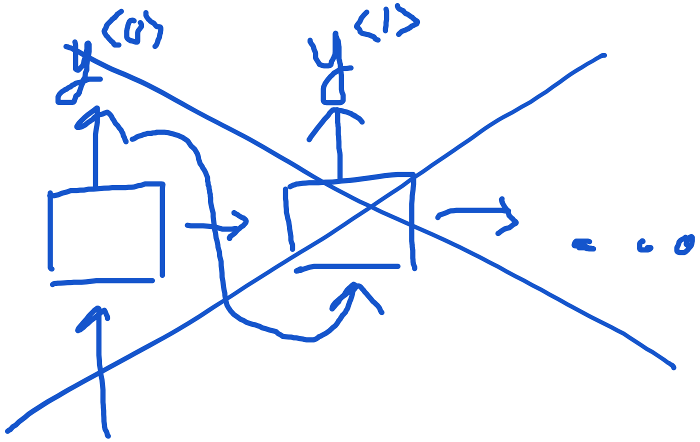
Transformers Intuition

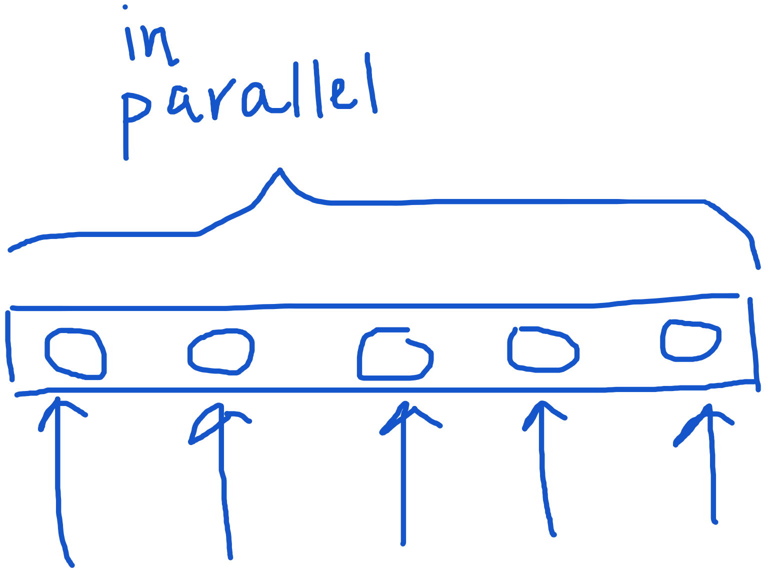
- Attention + CNN

Goal of self-attention: a sentence of five words will end up computing five representations for the five words (A1, A2, A3, A4, A5). And this will be an attention-based way computing representations for all the words in your sentence in parallel.

  - Self-Attention
  - Multi-Head Attention

Basically a forloop over the self-attention process.





CNN

[Vaswani et al. 2017, Attention Is All You Need]

deeplearning.ai

Sequence to sequence models

Let’s talk about the self-attention mechanism of trnasformers. Understand the core idea behind what makes transformer networks work.

Self-Attention

You have seen how attention is used with sequential neural networks such as RNN. To use attention with a style, you need to calculate self-attention, where you create attention-based representation for each of the words in the input sentences.

Self-Attention Intuition

$A(q, K, V)$= attention-based vector representation of a word

calculate for each word

Previously, we learned about word embeddings: one way to represent "l'afrique" would be to look up the word embbedings for l'afrique. But depending on the context, we can think of l'afrique as a site of historical interests or as a holiday destination, or as the world's second largest continent.

RNN Attention

$$\alpha^{<\cancel{t,t'}>} = \frac{\exp(e^{<t,t'>})}{\sum_{t'=1}^{T_x} \exp(e^{<t,t'>})}$$

Transformers Attention

Depending on how you think, you may choose to represent it differently and that is what A(3), this representation, will do.

$$A(q, K, V) = \sum_i \frac{\exp(q \cdot k^{<i>})}{\sum_j \exp(q \cdot k^{<j>})} v^{<i>}$$

It will look at the surrounding words to try to figure out what's actually going on in how we are talking about Africa in his sentence, and find the most appropriate representation for this. In terms of actual calculation, it is similar to RNN but differently to RNN, we will compute the representations in parallel for all 5 words in a sentence.

Attention-based representation for each word in input sentence

$x^{<1>}$	$x^{<2>}$	$x^{<3>}$	$x^{<4>}$	$x^{<5>}$
Jane	visite	l'Afrique	en	septembre

[Vaswani et al. 2017, Attention Is All You Need]

The key advantage of this representation A3=A(Q3, K, V) is the word 'l'Afrique' isn't some fixed word embedding.

Self-Attention

Instead, it lets the self-attention mechanism realize that l'Afrique is the destination of a visite.

softmax

$$A(q,K,V) = \sum_i \frac{\exp(e^{<q\cdot k^{<i>}>})}{\sum_j \exp(e^{<q\cdot k^{<j>}>})} v^{<i>}$$

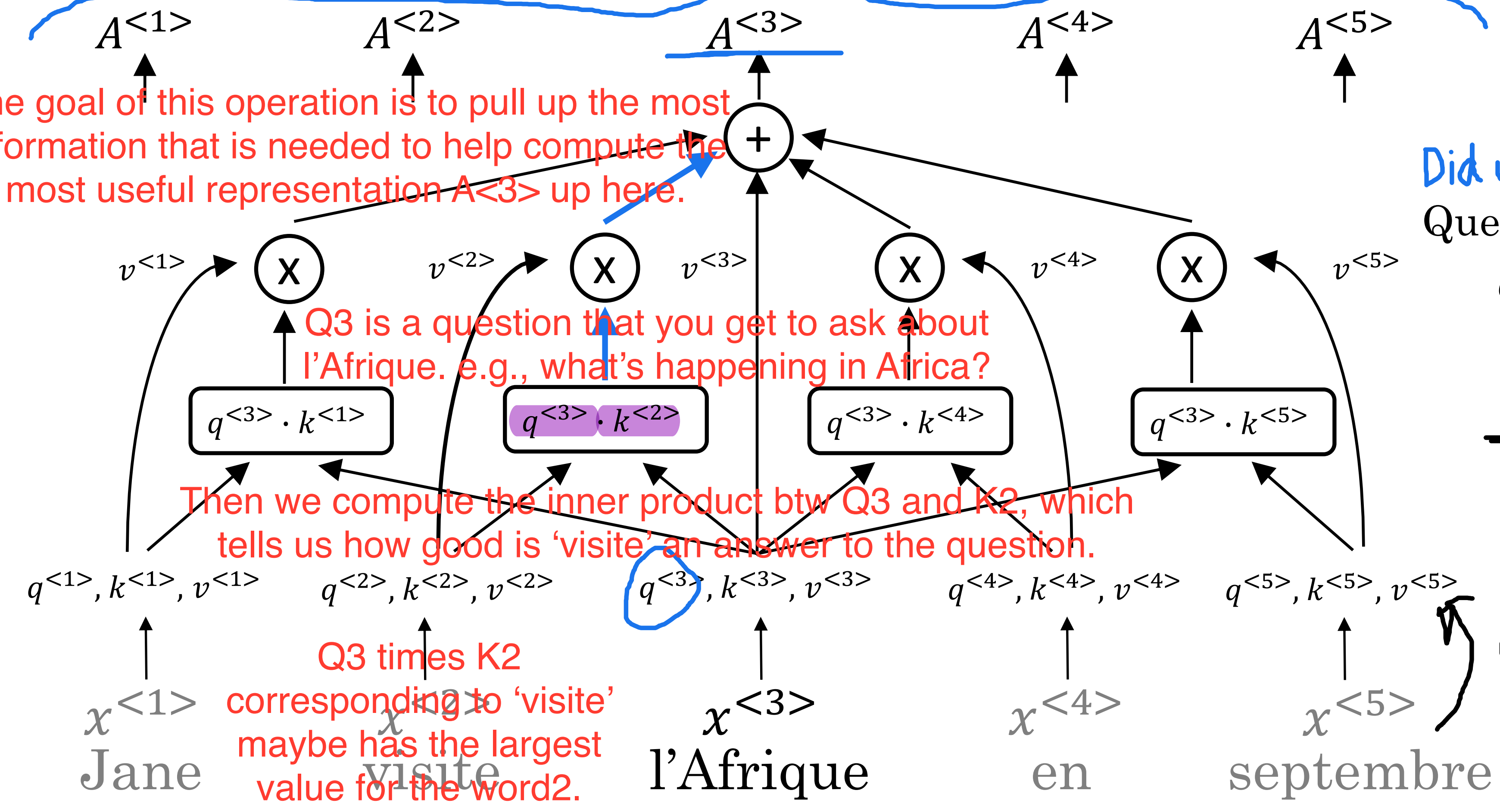
query (q), key (K), value (V)

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

This type of attention is also called the scaled dot-product attention

thus compute a richer, more useful representation for this word.

If X3 is word embedding for l'Afrique, then the way Q3 is computed is as a learned matrix. These matrices WQ, WK WV are parameters of this learning algorithm and they allow you to pull off these query, key, and value vectors for each word.



Did what?

Query (Q)	Key (K)	Value (V)
$q^{<1>}$	$k^{<1>}$	$v^{<1>}$
$q^{<2>}$	$k^{<2>}$	$v^{<2>}$
$q^{<3>}$	$k^{<3>}$	$v^{<3>}$
$q^{<4>}$	$k^{<4>}$	$v^{<4>}$
$q^{<5>}$	$k^{<5>}$	$v^{<5>}$

What's happening there?

person

action

Jane

visit

W^Q, W^K, W^V

q<3> = W^Q . X<3>
k<3> = W^K . X<3>
v<2> =W^V . X<3>

[Vaswani et al. 2017, Attention Is All You Need]

Andrew Ng

deeplearning.ai

Sequence to sequence models

Multi-Head Attention

W_2^Q, W_2^K, W_2^V: 2nd question: “When?” —> september

Multi-Head Attention

multiplied by A matrix W

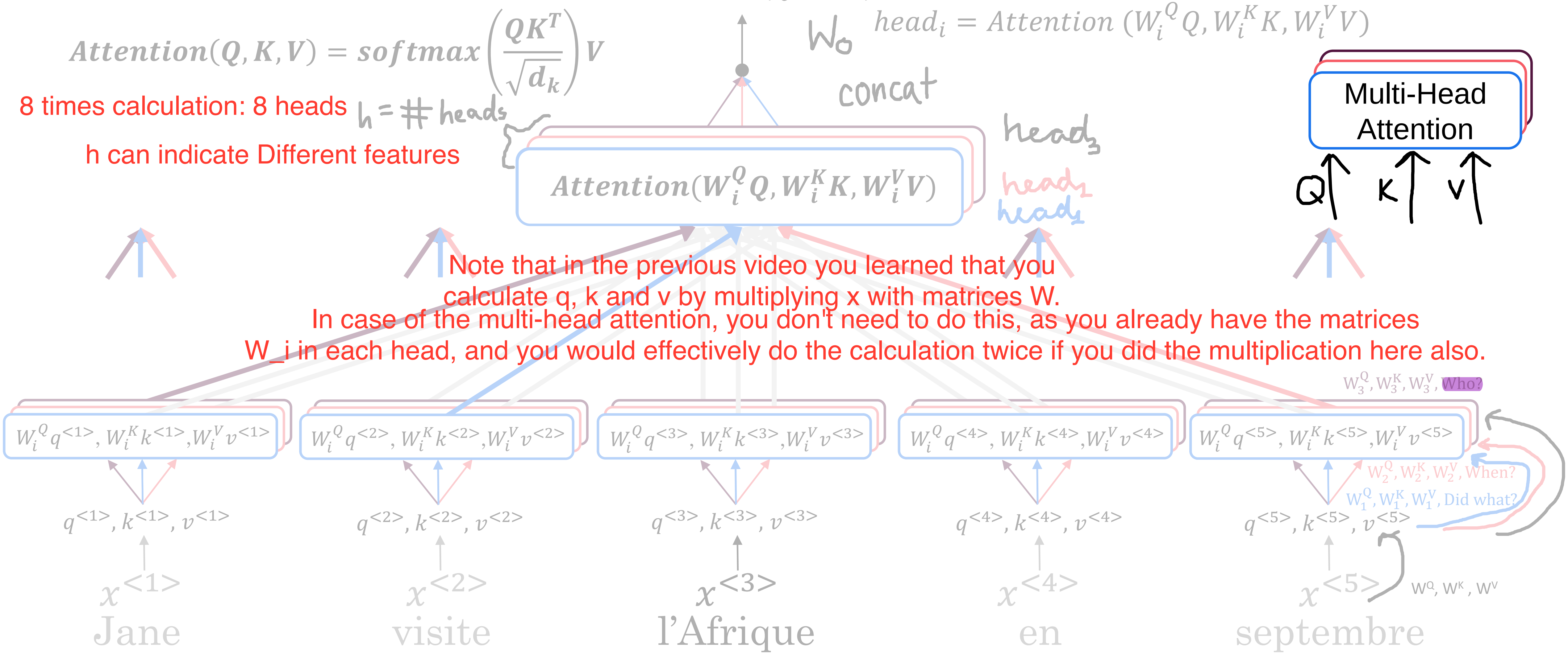
$$MultiHead(Q,K,V) = concat(head_1 head_2 \dots head_n)W_o$$

$$head_i = Attention\,(W_i^Q Q, W_i^K K, W_i^V V)$$

$$Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

8 times calculation: 8 heads

h can indicate Different features



Note that in the previous video you learned that you calculate q, k and v by multiplying x with matrices W.
In case of the multi-head attention, you don't need to do this, as you already have the matrices W_i in each head, and you would effectively do the calculation twice if you did the multiplication here also.

[Vaswani et al. 2017, Attention Is All You Need] W_1^Q, W_1^K, W_1^V - 1st question is “What is happening?”

In the simplest case of multi-headed self-attention you would actually use q=k=v=x. The reason we anyway show q, k and v in the previous slide as different values is that in one part of the transformer (where you calculate the attention between the input and output) the q, k and v are not all the same, as they carry different information.

Sequence to sequence models

The first step in the transformer is, the embeddings get fed into an encoder block which has a multi-head attention layer. This is exactly what we saw in the previous slide, where we feed in the values Q, K, and V computed from the embeddings and the weight matrices W.

This layer then produces a matrix that can be passed into a feed-forward neural network which helps determine what interesting features there are in the sentence.

Transformers

IN the transformer paper, this encoding block is repeated n times and a typical value for n is six.

Then, we will feed the output of the encoder into a decoder block.

deeplearning.ai

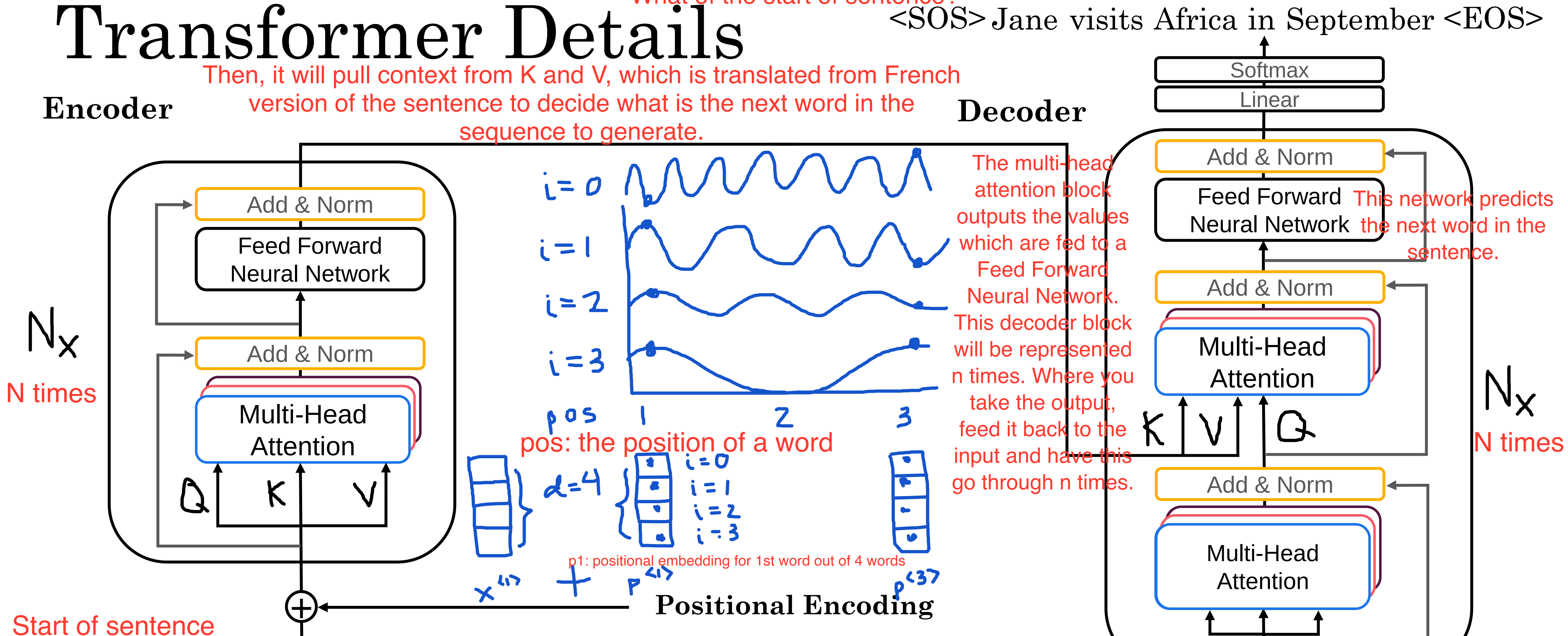
The decoder blocks' job is to output the english translation. The first output will be the start of sentence token. At every step, the decoder block will input the first few words, whatever we've already generated of the translation. The SOS token gets fed in to this multi-head attention block and just this one token, SOS token, is used to compute Q, K, and V for this multi-head attention block.

This first block's output is used to generate the Q matrix for the next multi-head attention block and the output of the encoder is used to generate K and V. Here's a second multi-head attention block with inputs Q, K and V as before.

Why is it structured this way? Maye the input down here is what you've translated of the sentence so far. This will ask a query to say, "What of the start of sentence?"

Transformer Details

Then, it will pull context from K and V, which is translated from French version of the sentence to decide what is the next word in the sequence to generate.



Positional Encoding

Start of sentence

$<SOS>\ x^{<1>}\ x^{<2>}\ \dots\ x^{<T_x-1>}\ x^{<T_x>}\ <EOS>$

Jane visite l'Afrique en septembre

$$PE_{(pos,2i)} = sin\left(\frac{pos}{1000^{\frac{2i}{d}}}\right)$$

$$PE_{(pos,2i+1)} = cos\left(\frac{pos}{1000^{\frac{2i}{d}}}\right)$$

k = 2i = 0
k =2i+ 1 = 1
k = 2i = 2
k = 2i+1 =3

$<SOS>\ y^{<1>}\ y^{<2>}\ \dots\ y^{<T_y-1>}\ y^{<T_y>}$

<SOS> Jane visits Africa in September

Note that in the video every value of i repeats twice. This is because you use each value of i to encode two dimensions using the sine and cosine functions (same i is used both for sine and cosine). To calculate i, you can use a helper index k, which is simply counting over the dimensions of the word embedding from 0 to d−1 and then calculate i as integer division of k//2. You will see this in practice during this weeks assignment.

[Vaswani et al. 2017, Attention Is All You Need]

Andrew Ng