



# Attention Is All You Need

Presented By-

Ankit Sharma (I77205)

Ashutosh Kumar (I77208)

Mohit Talreja (I77237)

Yash Jain (I77268)

# Background - Deep Learning for NLP

## What is NLP ?

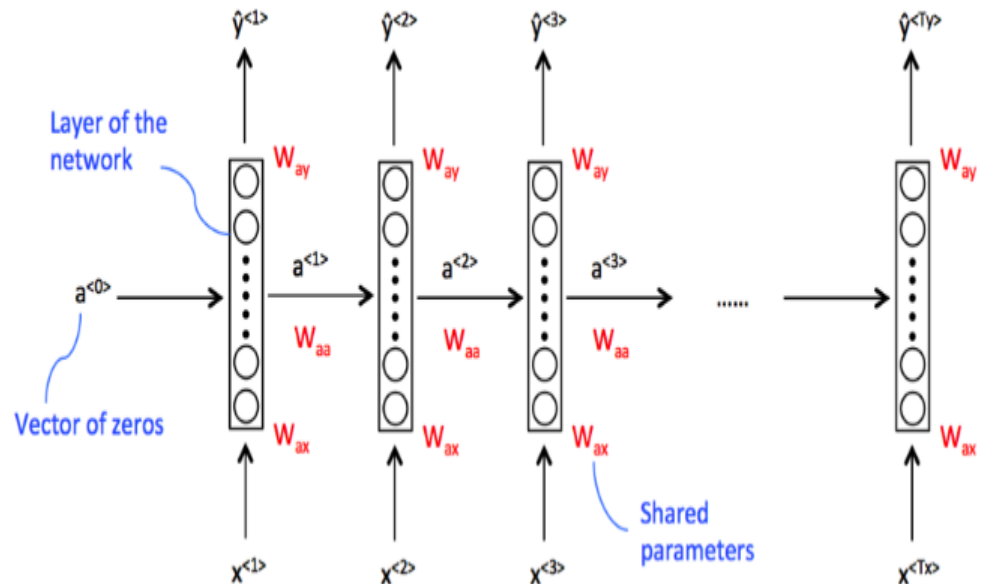
- Natural Language Processing (NLP) gives the machines the ability to read, understand and derive meaning from human languages.
- Fields- translation, speech recognition ,sentiment analysis etc.
- An active and challenging topic of research

## RNNs for NLP problems

- Deep Learning, especially RNNs have transformed NLP
- State-of-the-art across many tasks
- Recently, translation has been made easier by RNNs

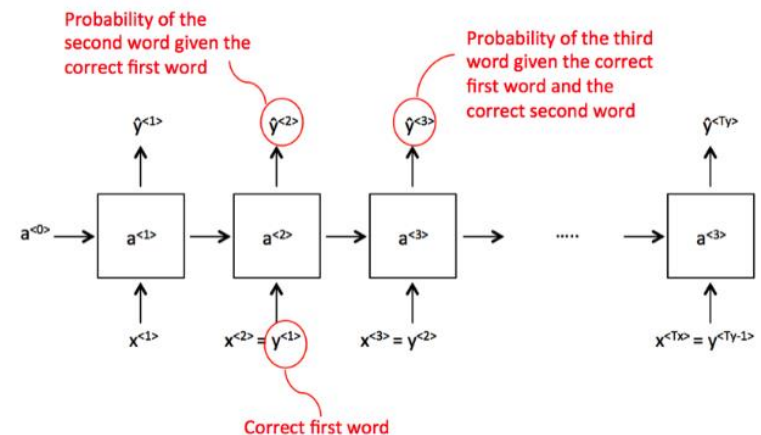
# Background - What are RNNs ?

- Type of Neural Network where the outputs from previous step are fed as input to the current step.
- The RNN scans through the data from left to right and the parameters used for each time step ( $W_{ax}$ ) are shared.
- The horizontal connections are governed by  $W_{aa}$  parameters, which are the same for every time step.
- The  $W_{ay}$ 's are the parameters that govern the output predictions.



# Background – RNNs in Translation

- $a_{<1>}$  makes a softmax prediction to try to figure out what is the probability of the first word  $\hat{y}_{<1>}$ .
- Then in the second step,  $a_{<2>}$  makes a softmax prediction given the correct first word ( $y_{<1>}$ ).
- All of the following steps make a prediction based on the correct words that come before them.



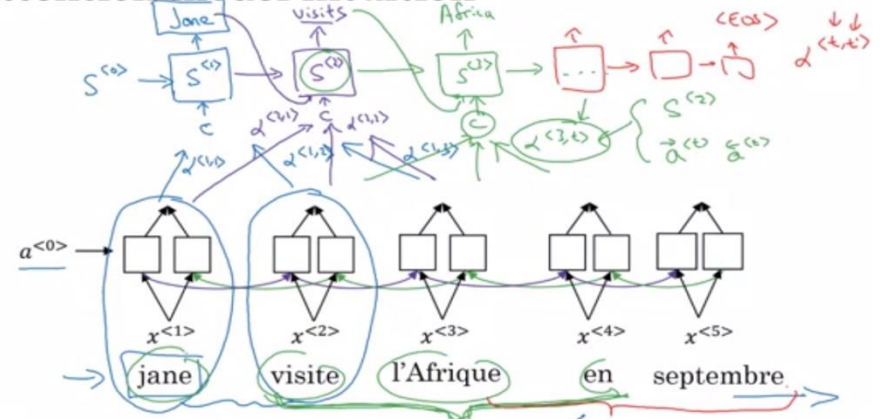
# Background - Disadvantages of RNN

- RNN generates hidden states (bottleneck) and not good at capturing long-range dependencies
- Hard to implement parallelization efficiently due to sequential nature, especially in long sequences.
- Backpropagation through long sequences is costly
- Transmitting local and global information

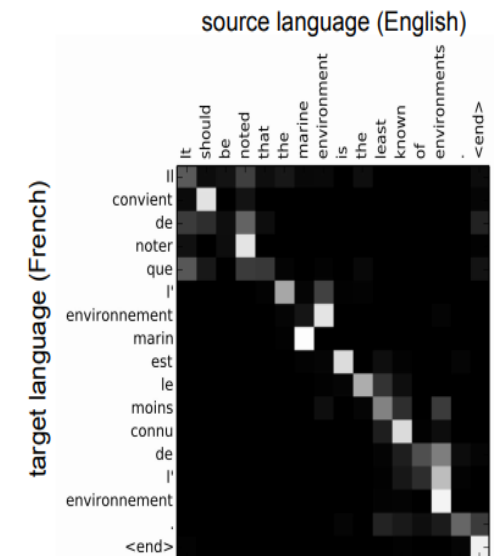
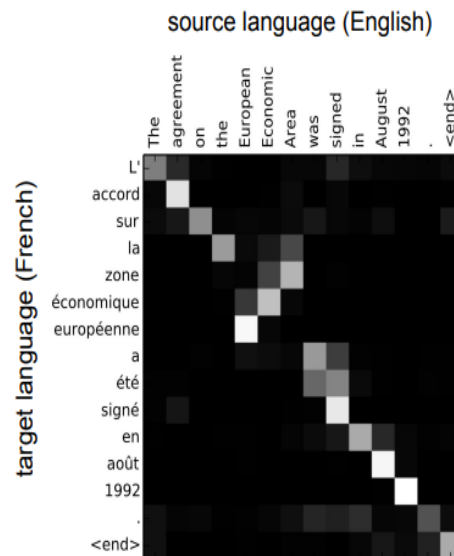
# Intuition of attention model

- Model takes into account the amount of attention that needs to be paid on any word in input sentence while translating any word.

Attention model intuition

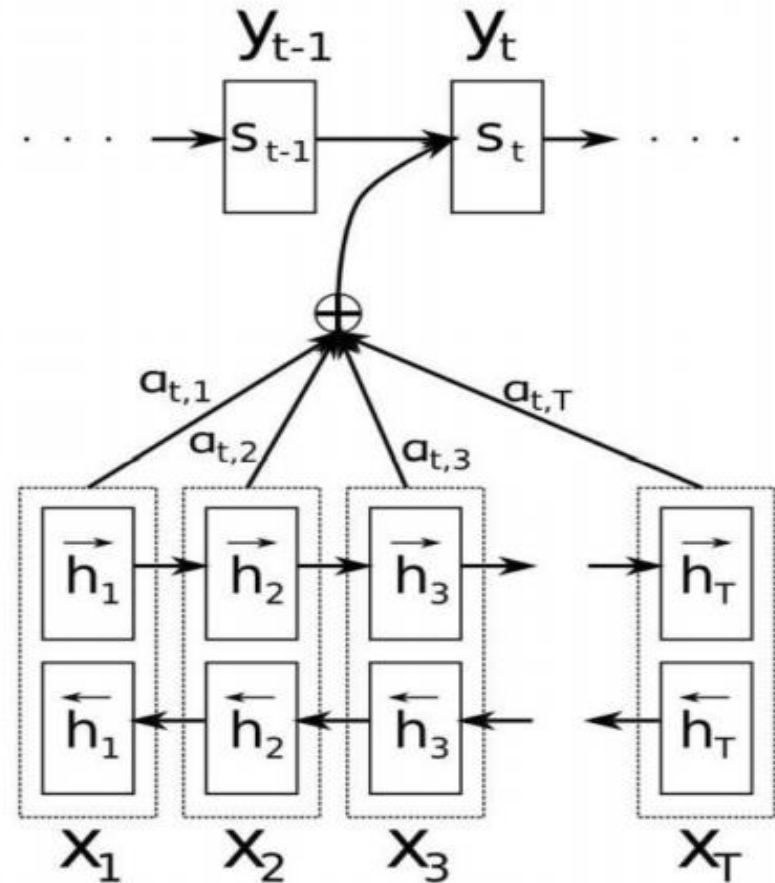


- $a(t, t')$  represents how much attention needs to be paid on  $t'^{\text{th}}$  word while calculating context for  $t^{\text{th}}$  word.



# How is attention model better than RNNs ?

- Removes bottleneck of RNN's Encoder-Decoder model and has ability to capture long-range dependencies.
- Provides context for given decoder step.





# Important Terminologies Explained



# Embedding

- Definition:

This step involves creation of D-dimensional dense vector with floating point values representing the strength of input words where words with similar meaning will be closer to each other.

In the transformer this step exists for both encoder ( for training)as well as decoder(for translation)

<b>cat</b> =>	1.2	-0.1	4.3	3.2
<b>mat</b> =>	0.4	2.5	-0.9	0.5
<b>on</b> =>	2.1	0.3	0.1	0.4
...			...	

*NOTE : The data has to be trained from scratch for embedding to work on random text*

## How it WORKS:

- The text imported are integer encoded and padded as per the length of the longest text.
- The Embedding layer takes the integer-encoded vocabulary and looks up the embedding vector for each word-index which then adds dimension to output array.

```
array([[7452, 1555, 7968, ..., 7984, 7976, 283],  
       [7567, 194, 3805, ..., 0, 0, 0],  
       [ 69, 57, 93, ..., 0, 0, 0],  
       ...,  
       [ 62, 347, 6073, ..., 0, 0, 0],  
       [ 10, 250, 1, ..., 0, 0, 0])
```

# How it WORKS

- A GlobalAveragePooling1D layer returns a fixed-length output vector which allows model to handle input of variable length, in the simplest way possible.
- The fixed-length output vector is then piped through dense hidden layers(of 16 units) and then through a sigmoid activation function to generate floating point between 0 and 1.

*NOTE : It is certainly advantageous over other such embedding as it generate dense embedding and floating value is useful for next step.*

# POSITIONAL ENCODING

- This step processes the input to generate words that are also closer to each other in relative position .This step is carried out for encoder and decoder .
- This encoding is a vector of sines and cosines at each position 'i', where each sine-cosine pair rotates at a different frequency.

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

- The vector from the previous step is fed to the above formula index-wise to generate rotational frequency where words closer in relative position will have similar frequency

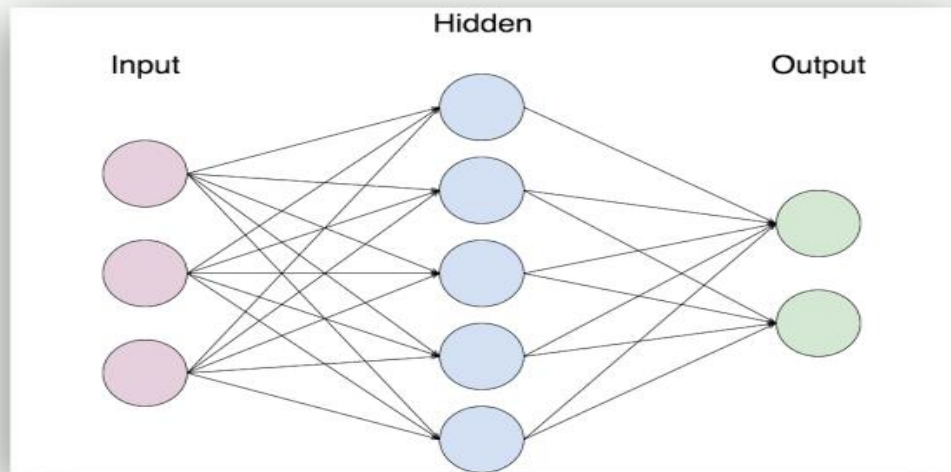
# MASKING

- It ensures that the model does not treat padding of step 1 as the input .The mask indicates where pad value 0 is present: it outputs a 1 at those locations, and a 0 otherwise.
- The second feature this steps adds is look ahead mask which masks future tokens(words) such that a given word is only predicted on basis of previous words

# FFN - Feed Forward Neural Network

It is one half of the conventional neural network that is made up of two dense layer with ReLU activation in between.

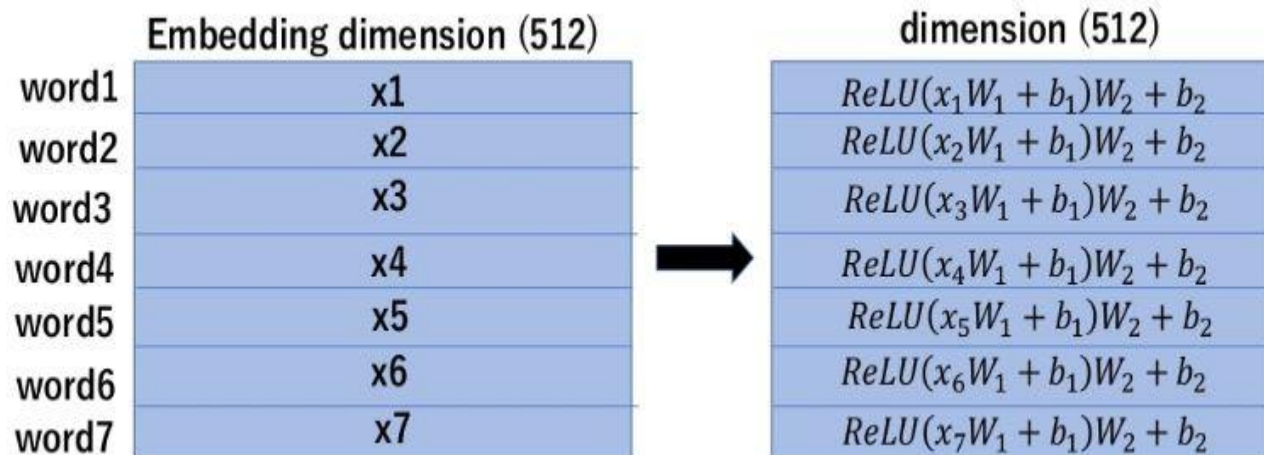
A sample model



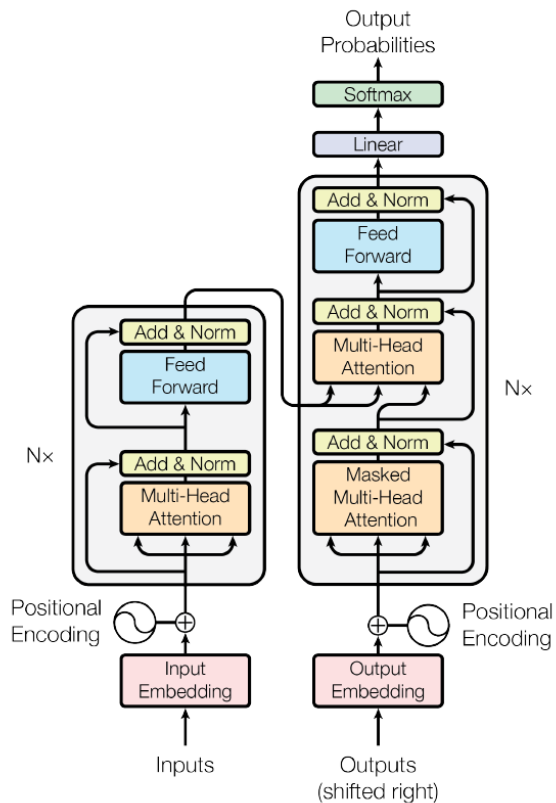
## FFN continued

- After calculating the multi-head attention and generating the concatenated vector the FFN is applied to each position separately

$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$



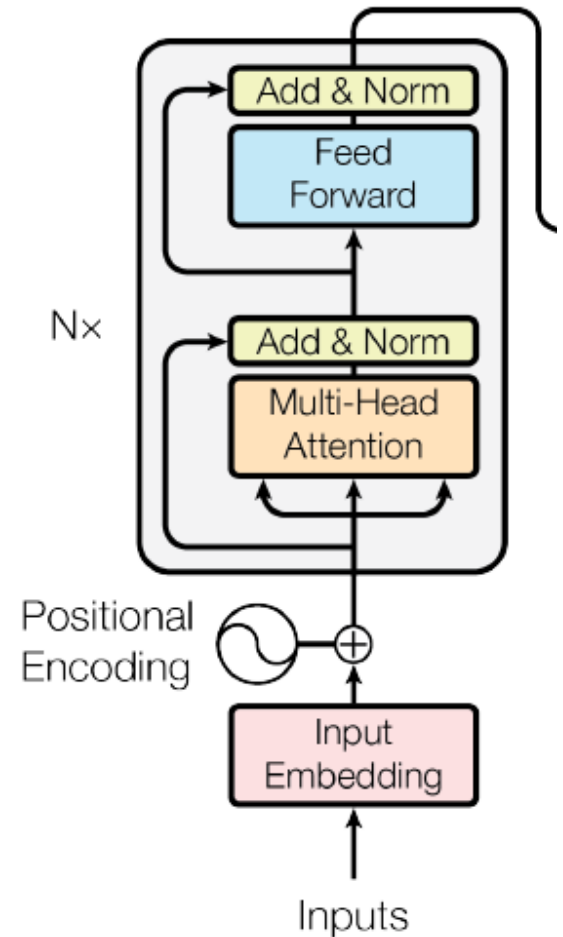
# The Transformer Architecture





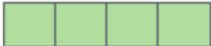
# The Encoder

- ❖  $N=6$  identical layers
- ❖ A layer comprises primarily of 2 sublayers: Self-Attention, Feed Forward Neural Network

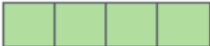


# Step 1: Embedding

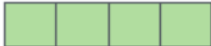
- Each input word is turned into a vector using an embedding algorithm. The size of this vector is 512.
- The size of this list is hyperparameter we can set – basically it would be the length of the longest sentence in our training dataset.

x<sub>1</sub> 

Je

x<sub>2</sub> 

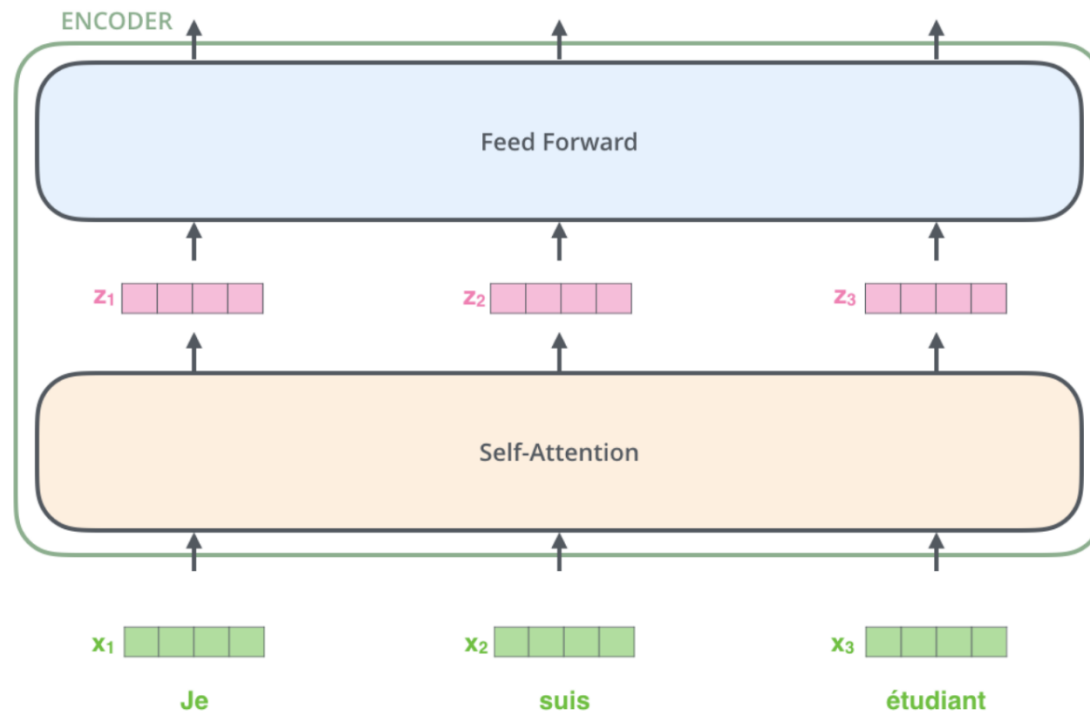
suis

x<sub>3</sub> 

étudiant

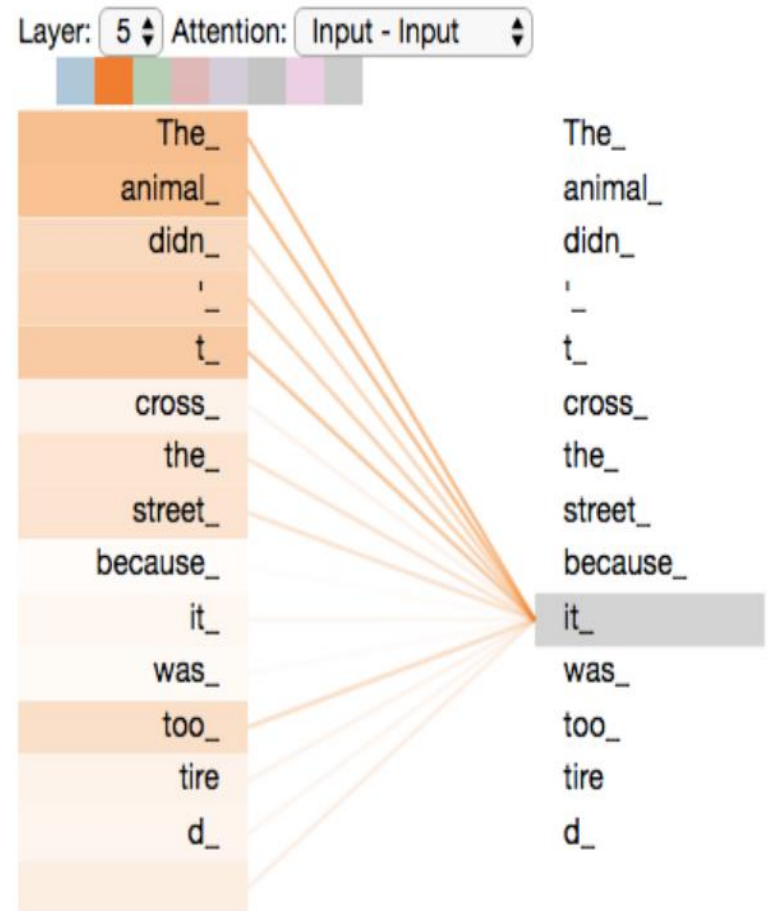
# Step 2: The Flow

- Each word has its own path.
- Dependencies between words exist only in the Self-Attention sublayer and not in the FFNN sublayer.
- Parallel execution possible.



# Step 3: Attention!!!

- An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.
- “The animal didn't cross the street because it was too tired”
- As the model processes each word (each position in the input sequence), self attention allows it to look at other positions in the input sequence for clues that can help lead to a better encoding for this word.



# Step 3.a: Scaled Dot-Product Attention

- The transformer views the encoded representation of the input as a set of **key-value** pairs, both of dimensions = input sequence length.
- In the context of Neural Machine Translation, both the keys and values are the encoder hidden states.
- In the decoder, the previous output is compressed into a **query** and the next output is produced by mapping this query and the set of keys and values.

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

Q = Query Matrix

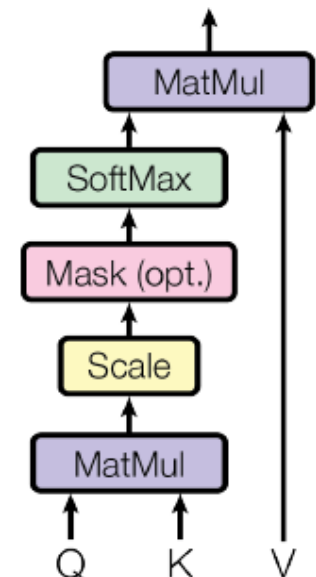
K = Key Matrix

V = Value Matrix

$d_k$  = dimensions of input matrices

$d_v$  = dimension of values matrix

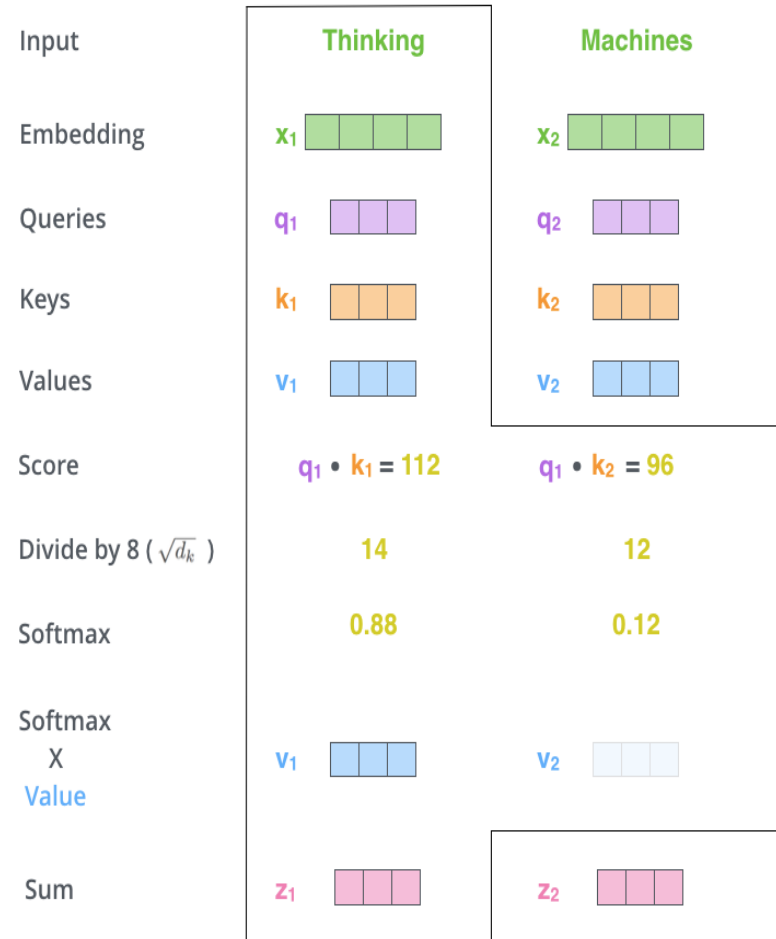
Softmax function to obtain the weights on the values.



# An Intuition to What Happens Atomically

1. Create three vectors from each of the encoder's input vectors. These vectors are created by multiplying the embedding by three matrices that we trained during the training process, these are  $W^Q, W^K, W^V$
2. Score calculation
3. Divide by 8: For large values of  $d_k$ , the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients.
4. Softmax normalization
5. Multiply each value vector by the softmax score
6. Sum up the weighted value vectors

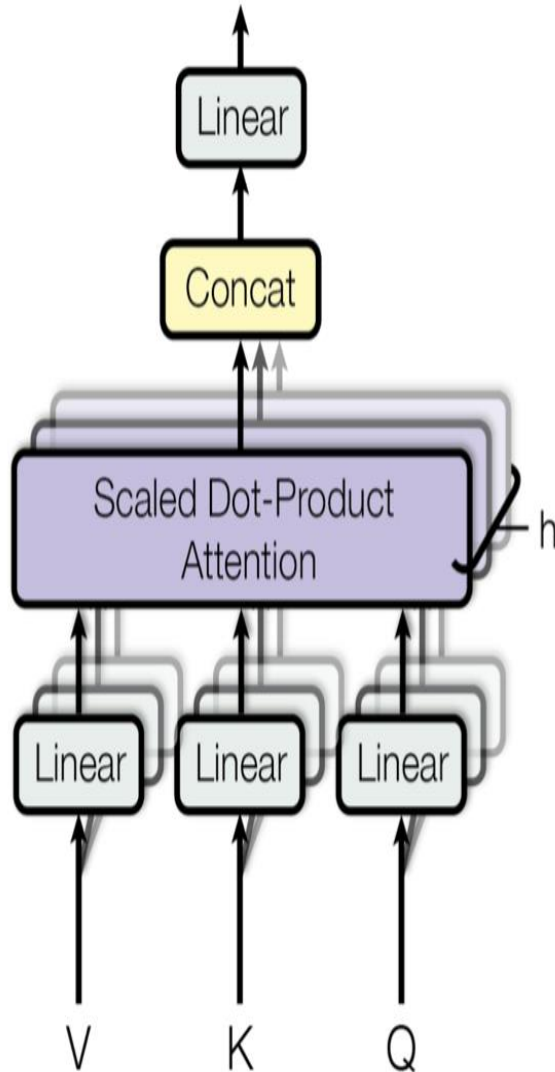
$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$





# Multihead-Attention

- Rather than only computing the attention once, the multi-head mechanism runs through the scaled dot-product attention multiple times in parallel.
- The independent attention outputs are simply concatenated and linearly transformed into the expected dimensions.



- $Q$ ,  $K$  and  $V$  are mapped into lower dimensional vector spaces using weight matrices and then the results are used to compute attention (the output of which we call a 'head').

$$\text{head} = \text{Attention}(QW^Q, KW^K, VW^V)$$

- We have  $h$  such sets of weight matrices which gives us  $h$  heads.

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

- The  $h$  heads are then concatenated and transformed using an output weight matrix.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

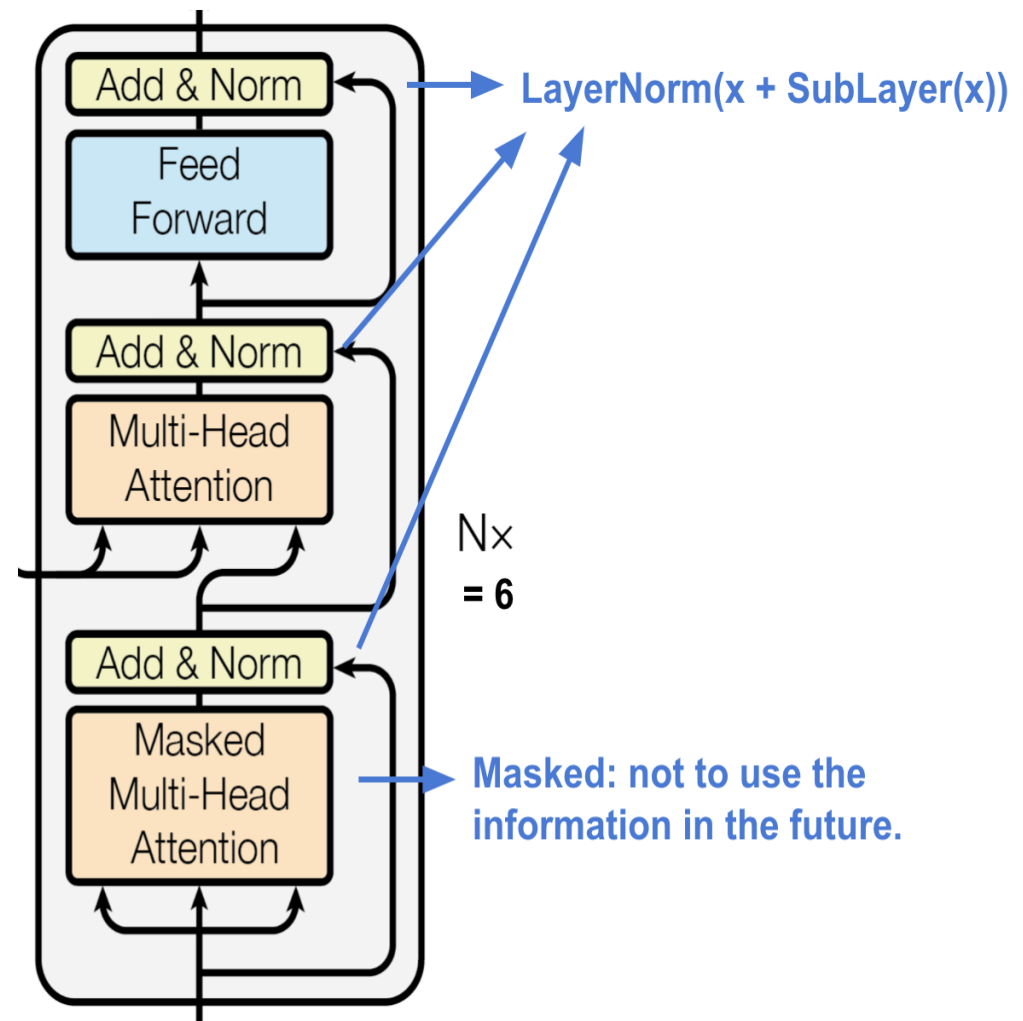


# The Decoder

The decoder is able to retrieval from the encoded representation.

- A stack of  $N = 6$  identical layers.
- Each layer has two sub-layers of multi-head attention mechanisms and one sub-layer of fully-connected feed-forward network.
- Similar to the encoder, each sub-layer adopts a residual connection and a layer normalization.

◦ The first multi-head attention sub-layer is modified to prevent positions from attending to subsequent positions, as we don't want to look into the future of the target sequence when predicting the current position.



# Training and Optimizing

- Training was carried out on English- German and English-French translation dataset.
- Optimization was done using Adam's Formula:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$

- Learning rate was increased linearly for the first 'warmup\_steps' training steps, and decreased thereafter proportionally to the inverse square root of the step number.
- warmup\_steps = 4000.

# Results

BLEU Score is an evaluation metric which represents how similar machine translation and actual translation are.

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	<b>41.29</b>	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	<b><math>3.3 \cdot 10^{18}</math></b>	
Transformer (big)	<b>28.4</b>	<b>41.8</b>	$2.3 \cdot 10^{19}$	

As it can be seen by the BLEU Scores, The model clearly outperforms the previous best reported models for translation, at a fraction of training cost.

# Model Variations

- To evaluate the importance of different components of the Transformer.
- Hyperparameters like no. of attention heads , attention key and value dimensions etc. were varied and change in performance of the model was noted.

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	$N$	$d_{\text{model}}$	$d_{\text{ff}}$	$h$	$d_k$	$d_v$	$P_{\text{drop}}$	$\epsilon_{ls}$	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$		
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65		
(A)					1	512				5.29	24.9			
					4	128				5.00	25.5			
					16	32				4.91	25.8			
					32	16				5.01	25.4			
(B)					16					5.16	25.1	58		
					32					5.01	25.4	60		
(C)	2									6.11	23.7	36		
	4									5.19	25.3	50		
	8									4.88	25.5	80		
		256			32	32				5.75	24.5	28		
		1024			128	128				4.66	26.0	168		
			1024							5.12	25.4	53		
			4096							4.75	26.2	90		
(D)							0.0			5.77	24.6			
							0.2			4.95	25.5			
								0.0		4.67	25.3			
								0.2		5.47	25.7			
(E)	positional embedding instead of sinusoids									4.92	25.7			
big	6	1024	4096	16					0.3	300K	4.33	26.4	213	

# Summary/Conclusion

- Here Transformer was presented, first sequence transduction model based entirely on attention, replacing recurrent layers appearing in encoder-decoder architectures with multi-headed self attention.
- Transformer can be trained significantly faster than architectures based on recurrent/convolutional layers( there is much less computation for training).
- Hence, Attention is All You Need!!

# Tools Used

## Libraries:

- Tensorflow
- TFDS dataset from  
TED\_HRLR\_TRANSLATE
- Numpy
- Matplotlib

## Platform:

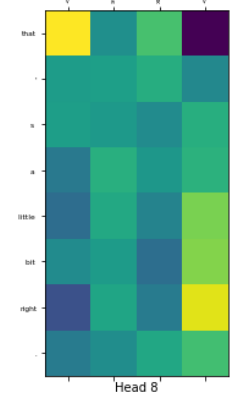
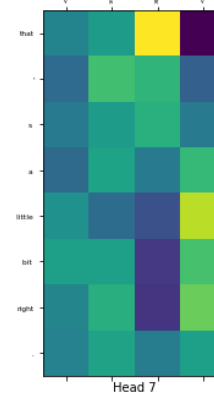
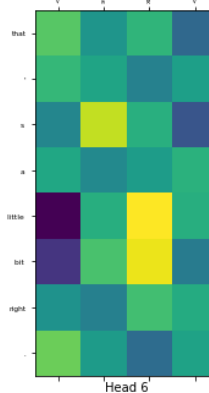
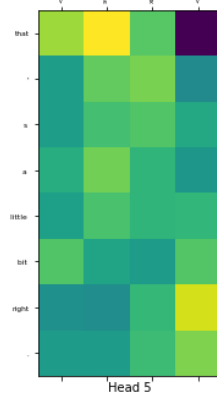
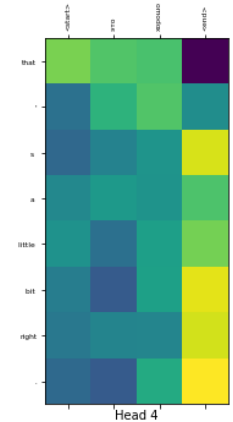
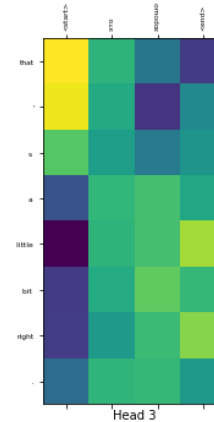
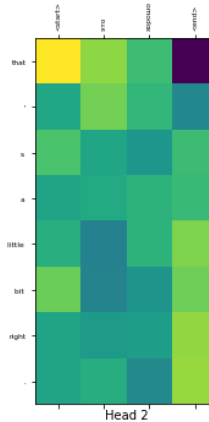
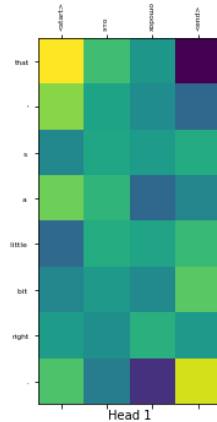
- Google Colab

# Results from our Code

Russian to English, EPOCHS = 1, Training examples 208,106

Input: это хорошо

Prediction: that 's a little bit right .



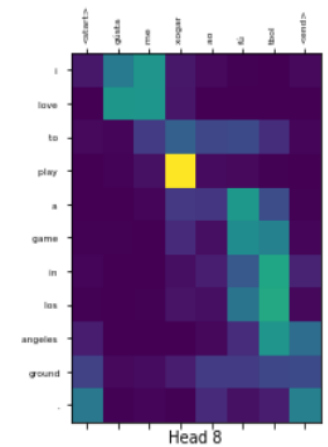
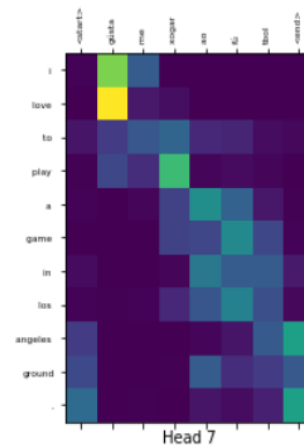
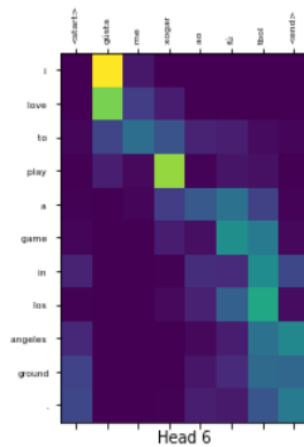
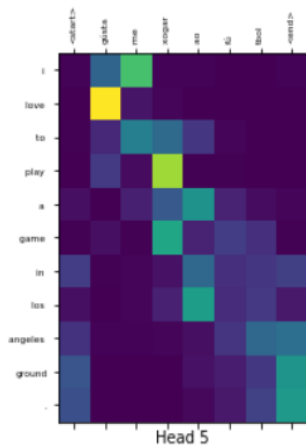
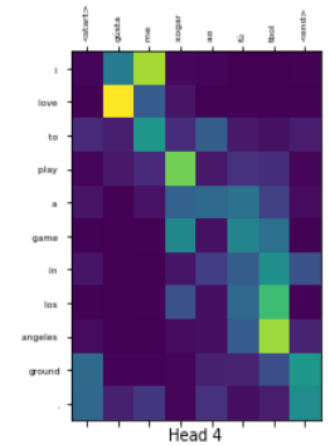
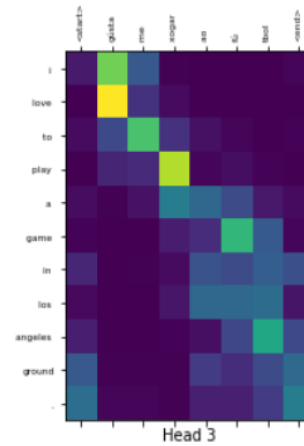
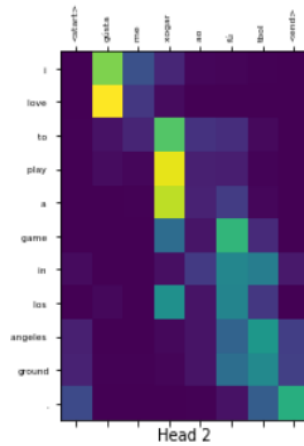
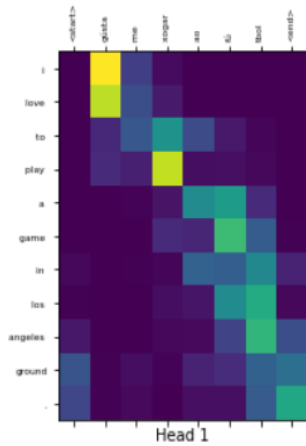
Real: That is good.



# Galician to English, EPOCHS = 30, Training examples = 10,017

Input: gústame xogar ao fútbol

Prediction: i love to play a game in los angeles ground .

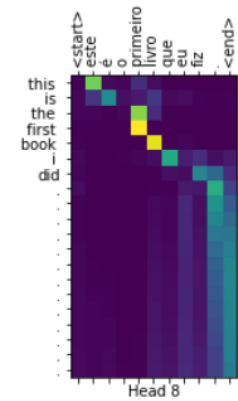
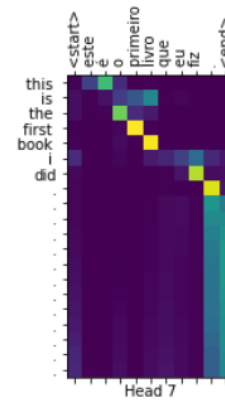
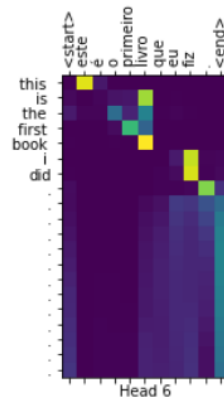
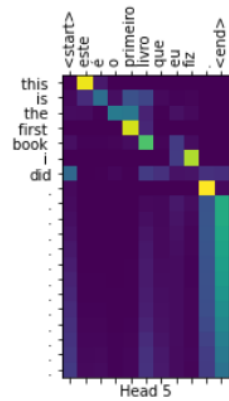
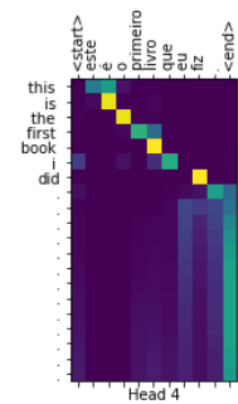
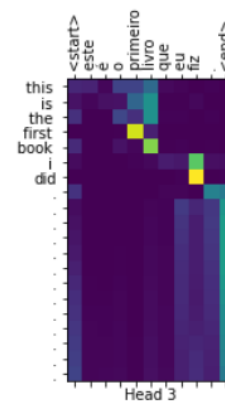
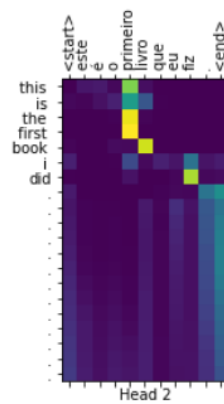
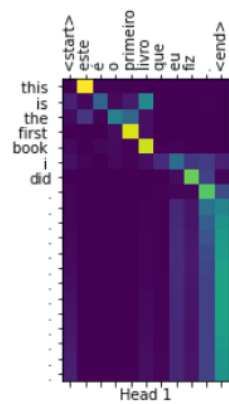


Real: I like to play football

# Portuguese to English, EPOCHS = 20, Training examples = 51,785

Input: este é o primeiro livro que eu fiz.

Predicted translation: this is the first book i did.....



Real translation: this is the first book i've ever done.