Transformer Anatomy

Attention is really all you need?

Speaker: Simeon Harrison
Trainer at EuroCC Austria

EURO
AUSTRIA

Transformer Anatomy

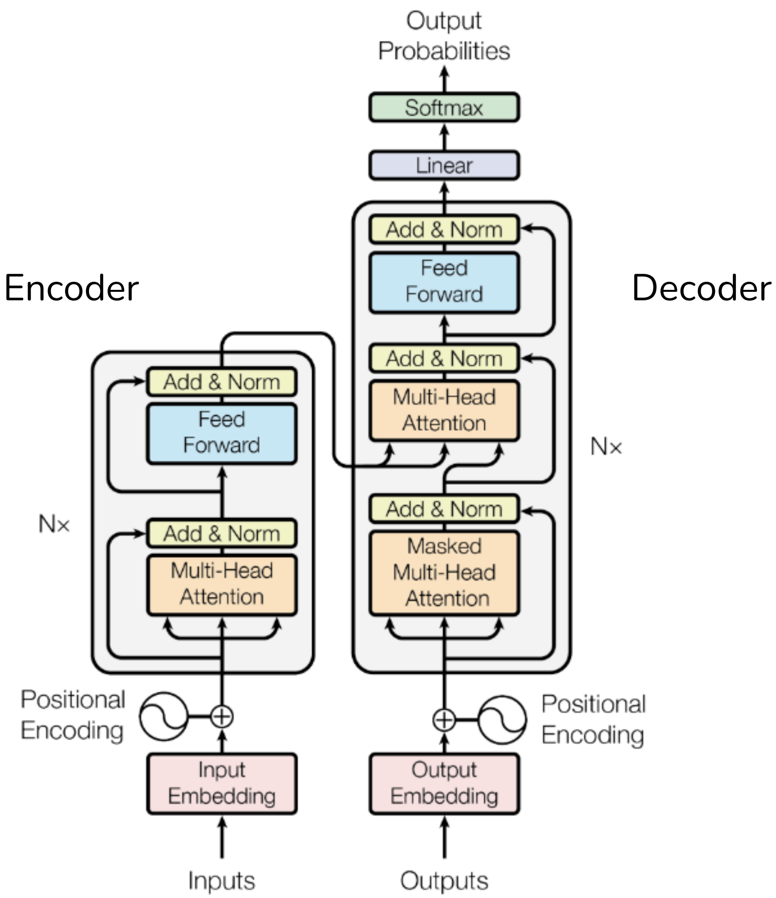
The original architecture

A transforer consists of a encoder and/or decoder block.

Words (tokens) are input as numerical representations (embeddings).

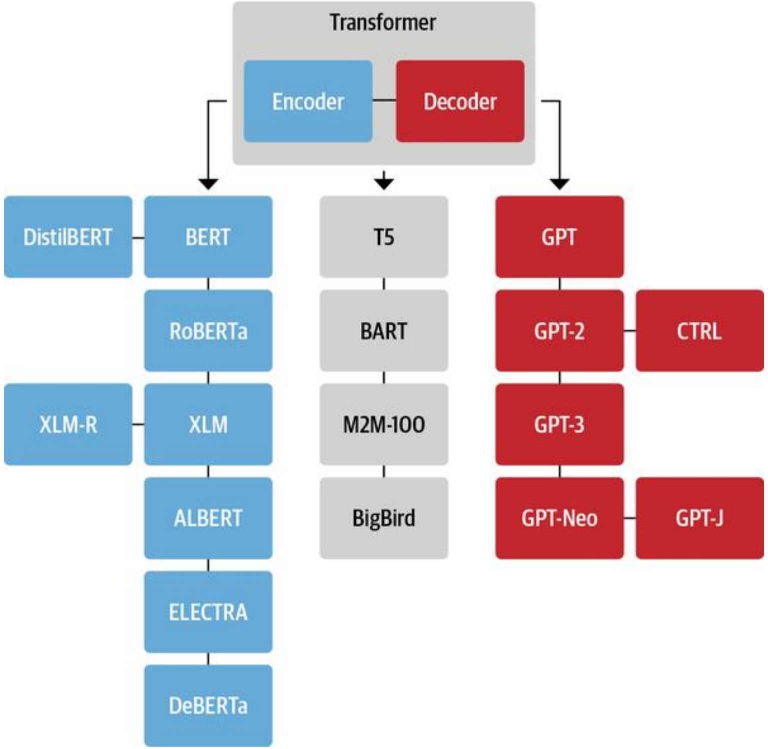
About 1/3 of all parameters are in the multi-head attention blocks

About 2/3 of all parameters are in the feed forward networks (also known as multi layer perceptron)



Source: “Attention Is All You Need”, Vaswani et al.

Transformer Family



Source: Natural Language Processing with Transformers, O'Reilly

Encoder only:
These models ecxel at text classification, named entity recognition, and question answering

Decoder only:
Very good at predicting the next word in a sequence, therefore mostly used for text generation

Encoder-Decoder:
These models are often used for machine translation or summerization tasks.

Context Is All You Need

Embeddings

Here, we will refer to “word” instead of “token”, as it makes the content easier to explain.

A word embedding comes as a multi dimensional vector (e.g. 12.000 dim).

The initial word embedding in all of the examples of the word „mole“ is the same.

The European mole is a mammal

Take a biopsy of the mole

6.02×10^{23}

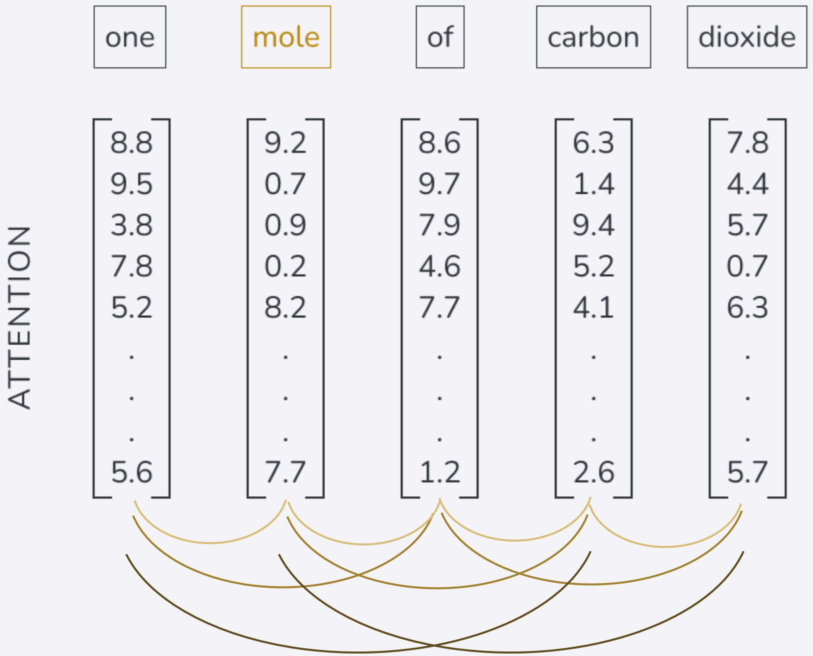
One mole of carbon dioxide

Context Is All You Need

Attention

The word „mole“ should be represented by a **unique vector** in the embedding space, depending on ist **context.**

An **attention** block should **compute the vectors that you need to add** to the original, generic vector to get it to the correct, meaningfull, rich representation, depending on the context in which the word is used.



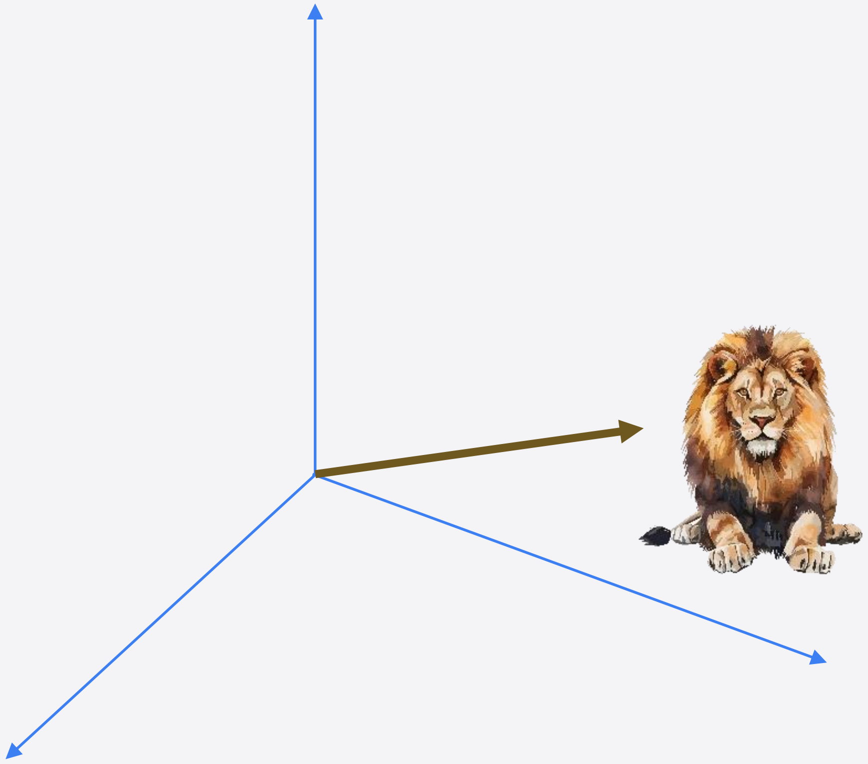
EURO AUSTRIA

Context Is All You Need

Lion

We associate the word „lion“ with a big cat, living wild on the African continent.
We probably imagine a majestic predator with a big mane.

The embedding of the word „lion“ is a vector with a certain length and direction within the embedding space.

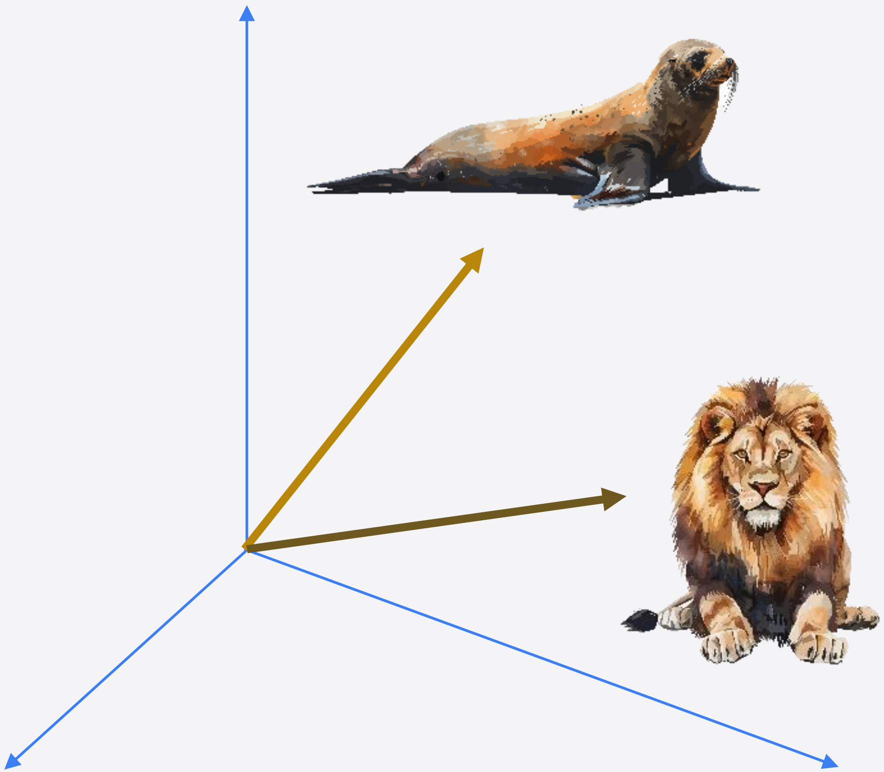


Context Is All You Need

Sea Lion

However, as soon we add the word „sea“ infront of „lion“ we imagine a totally different animal.

The same goes for the embedding. The attiontion mechanism needs to update the direction and length of the vector so that it represents the animal in question correctly.



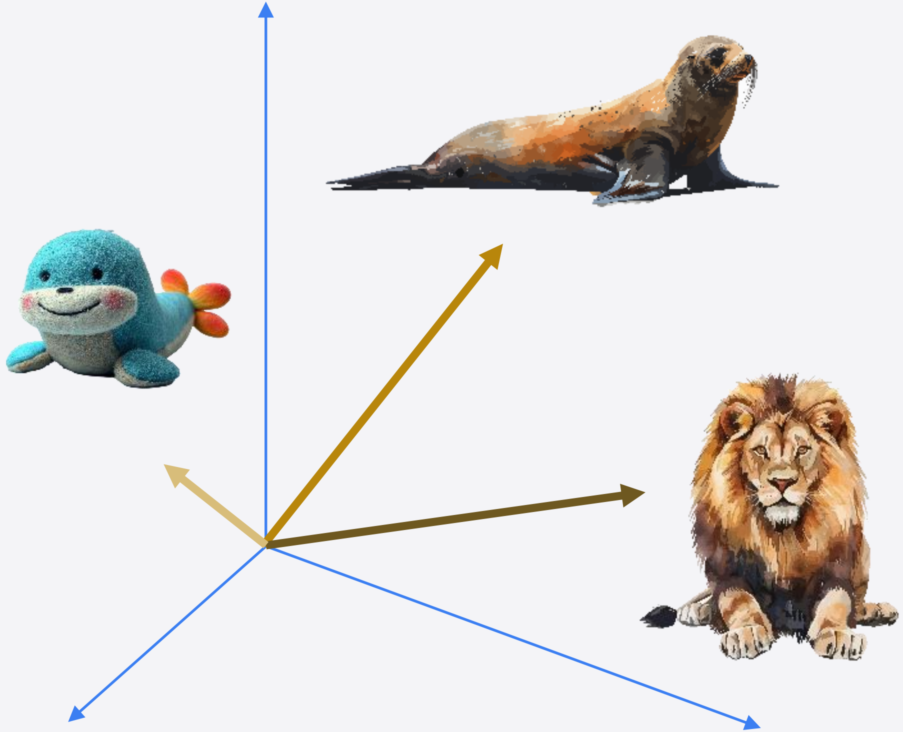
Context Is All You Need

Sea Lion Cuddly Toy

The context depends on more than just the immediate words to the left and right.

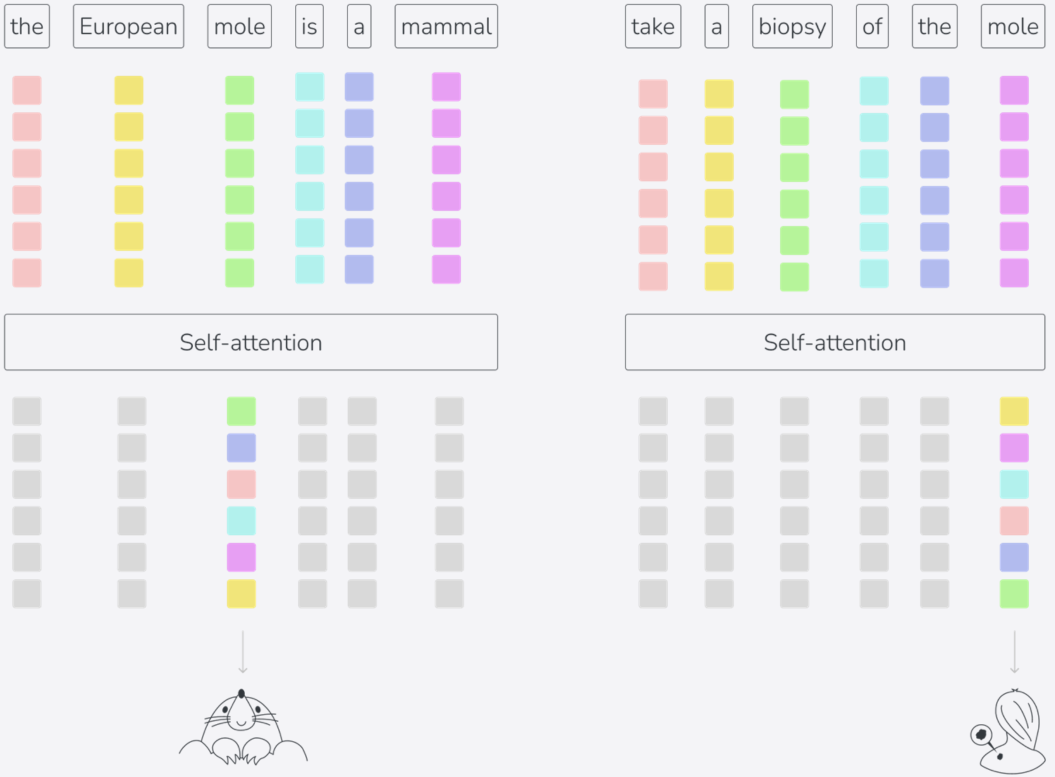
The embedding of „sea lion cuddly toy“ will certainly be very different of just „lion“.

In order to achieve that the vector for „lion“ needs to attend to all the other words in the input (context size).

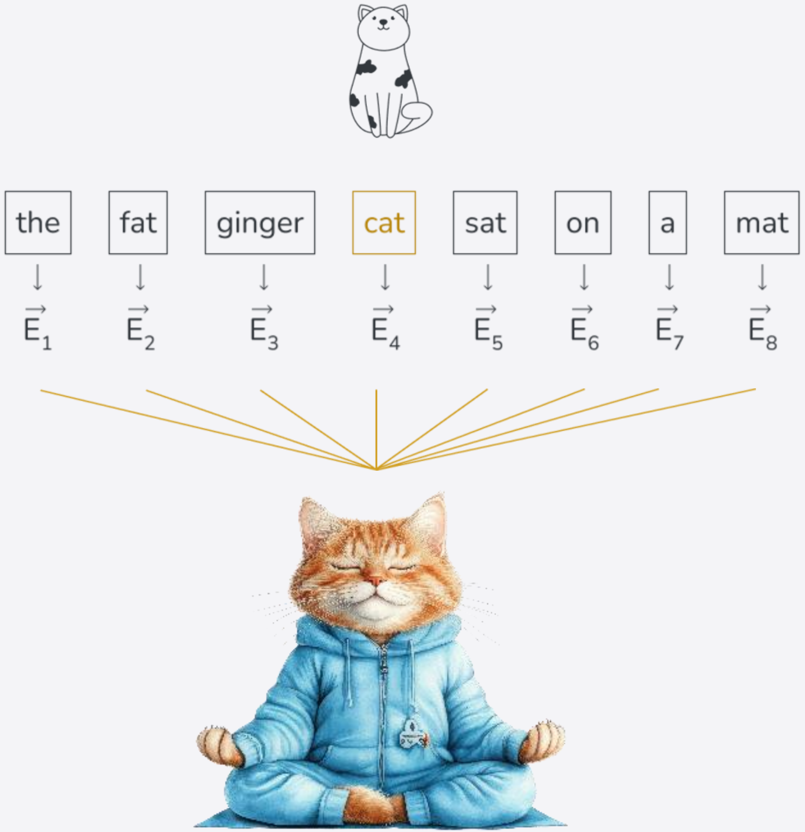


EURO AUSTRIA

Self Attention



Self Attention



Attention! Queries, Keys and Values

The Attention Score

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

Q.....query matrix

K.....key matrix

V.....value matrix

d.....dimension of (smaller) query-key space

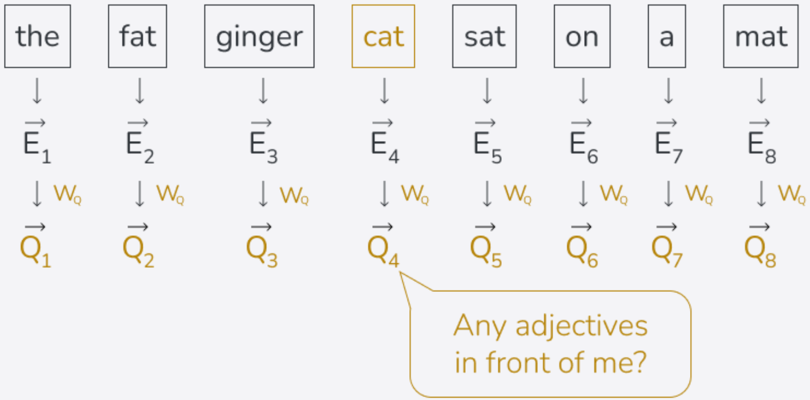
Attention! Queries, Keys and Values

Query

The query vector is obtained by multiplying a matrix full of trainable parameters with the embedding vector of that word (token).
Each word has its own query vector:

$$\vec{Q_i} = W_Q \vec{E_i}$$

It mappes the embedding vector to a much smaller dimensional space (e.g. 128 dimensions)



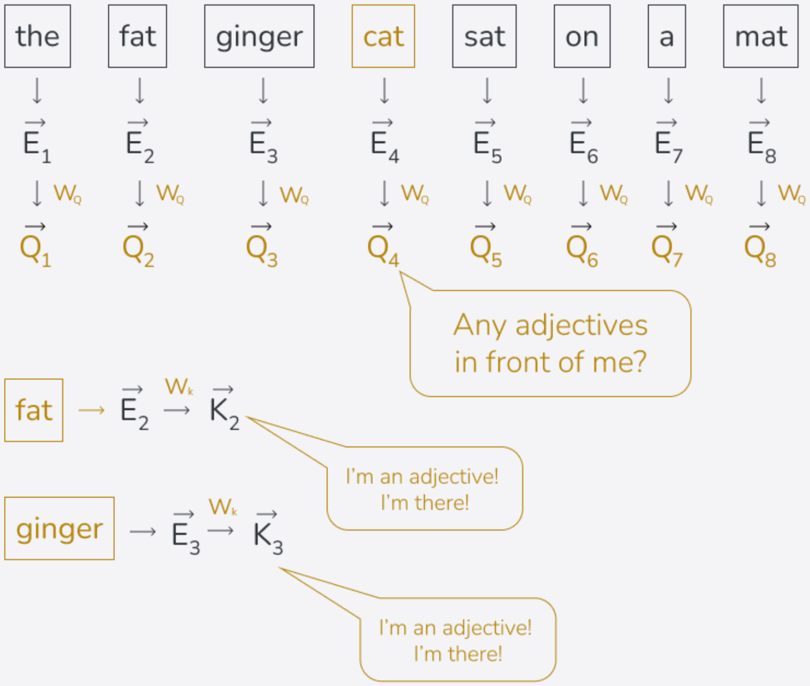
Imagine it like this:
One particular attention head is focussing on nouns. The query vector is like looking for labels with „adjective“ on them to better understand the noun.
In reality, this is much more abstract.

Attention! Queries, Keys and Values

Key

The key vector is also obtained by multiplying a matrix full of trainable parameters with the embedding vector of that word (token). Each word also has its own key vector: $\overrightarrow{K_i} = W_K \overrightarrow{E_i}$

Imagine it like this:
We are still dealing with the particular attention head that is focussing on nouns. The key is answering the question the query raised.

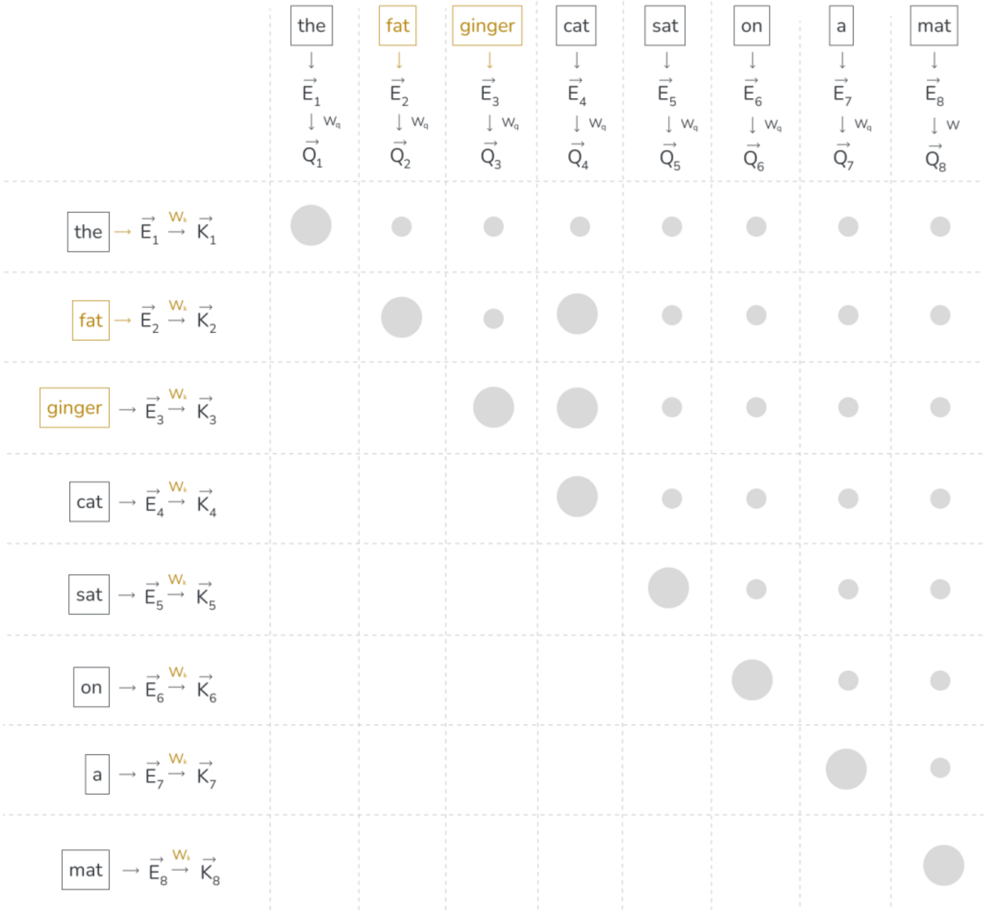


Attention! Queries, Keys and Values

Query – Key

Compute the dot product with each query-key pair, to determine how well the key matches the query. Where the queries and keys align, the dot product is larger.

Imagine it like this:
With our previous example, the dot product of the key vectors of „fat“ and „ginger“ with the vector of „cat“ yields the largest result. The embeddings of „fat“ and „ginger“ attend to „cat“.

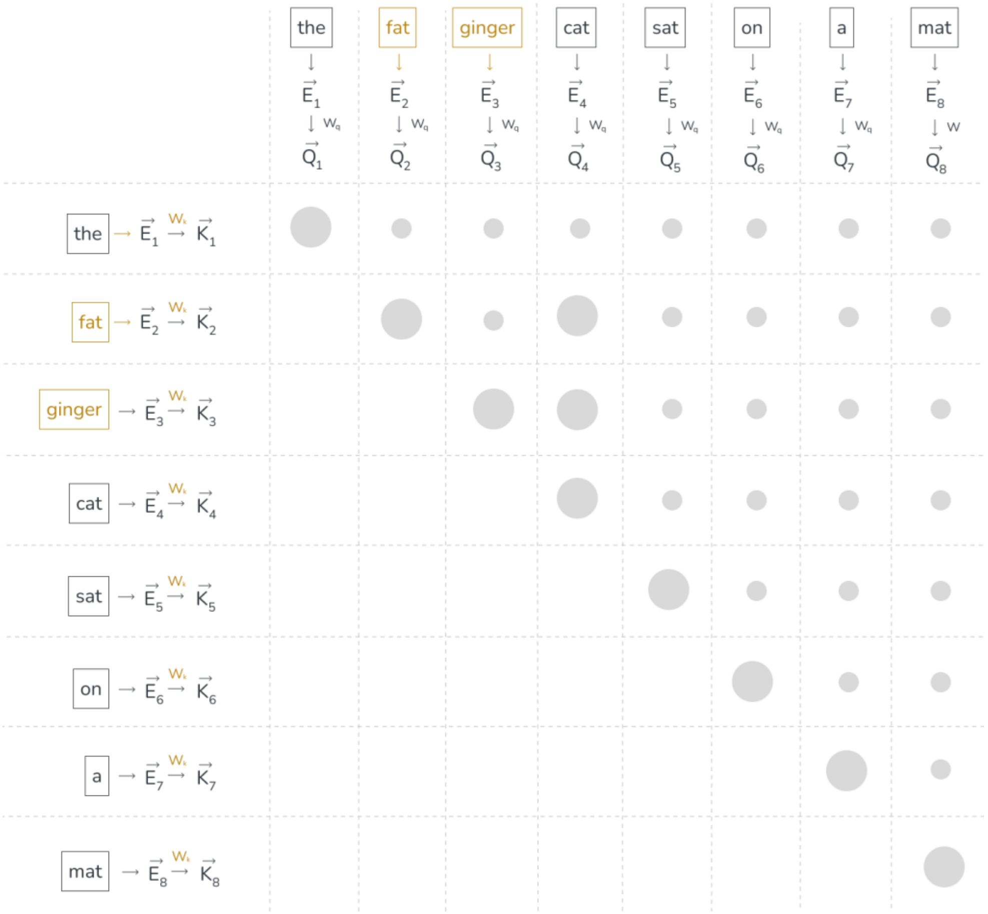


Attention! Queries, Keys and Values

Attention Pattern

Lower left dot products are masked, as we want the model to predict every next word during training. To prevent data leakage, future words should not influence previous ones.

The size of the attention pattern is the context size squared.
This is why the context size can be a substantial bottleneck.



Attention! Queries, Keys and Values

Attention

So far, we have determined which word is relevant to which other word (dot product of query and key).

We would like to use this as a score for how relevant every word is to update the meaning of other words.

For numerical stability the dot product is devided by the square root of the dimension of the query-key space.

To normalize the numbers to be between 0 and 1 we apply softmax.

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

EURO AUSTRIA

Attention! Queries, Keys and Values

Value

The value matrix multiplied by the embedding of the preceding word results in the value vector.

$$\vec{V_i} = W_K \vec{E_i}$$

	the	fat	ginger	cat	sat	on	a	mat
	$\vec{E}_1$	$\vec{E}_2$	$\vec{E}_3$	$\vec{E}_4$	$\vec{E}_5$	$\vec{E}_6$	$\vec{E}_7$	$\vec{E}_8$
$\vec{E}_1 \xrightarrow{W_v} \vec{V}_1$	$1.0\,\vec{V}_1$	$0.0\,\vec{V}_1$	$0.0\,\vec{V}_1$	$0.0\,\vec{V}_1$	$0.0\,\vec{V}_1$	$0.0\,\vec{V}_1$	$0.0\,\vec{V}_1$	$0.0\,\vec{V}_1$
	+	+	+	+	+	+	+	+
$\vec{E}_2 \xrightarrow{W_v} \vec{V}_2$	$0.0\,\vec{V}_2$	$1.0\,\vec{V}_2$	$0.0\,\vec{V}_2$	$0.42\,\vec{V}_2$	$0.0\,\vec{V}_2$	$0.0\,\vec{V}_2$	$0.0\,\vec{V}_2$	$0.0\,\vec{V}_2$
	+	+	+	+	+	+	+	+
$\vec{E}_3 \xrightarrow{W_v} \vec{V}_3$	$0.0\,\vec{V}_3$	$0.0\,\vec{V}_3$	$1.0\,\vec{V}_3$	$0.58\,\vec{V}_3$	$0.0\,\vec{V}_3$	$0.0\,\vec{V}_3$	$0.0\,\vec{V}_3$	$0.0\,\vec{V}_3$
	+	+	+	+	+	+	+	+
$\vec{E}_4 \xrightarrow{W_v} \vec{V}_4$	$0.0\,\vec{V}_4$	$0.0\,\vec{V}_4$	$0.0\,\vec{V}_4$	$0.0\,\vec{V}_4$	$0.0\,\vec{V}_4$	$0.0\,\vec{V}_4$	$0.0\,\vec{V}_4$	$0.0\,\vec{V}_4$
	+	+	+	+	+	+	+	+
$\vec{E}_5 \xrightarrow{W_v} \vec{V}_5$	$0.0\,\vec{V}_5$	$0.0\,\vec{V}_5$	$0.0\,\vec{V}_5$	$0.0\,\vec{V}_5$	$0.0\,\vec{V}_5$	$0.0\,\vec{V}_5$	$0.0\,\vec{V}_5$	$0.0\,\vec{V}_5$
	+	+	+	+	+	+	+	+
$\vec{E}_6 \xrightarrow{W_v} \vec{V}_6$	$0.0\,\vec{V}_6$	$0.0\,\vec{V}_6$	$0.0\,\vec{V}_6$	$0.0\,\vec{V}_6$	$0.99\,\vec{V}_6$	$1.0\,\vec{V}_6$	$0.0\,\vec{V}_6$	$0.0\,\vec{V}_6$
	+	+	+	+	+	+	+	+
$\vec{E}_7 \xrightarrow{W_v} \vec{V}_7$	$0.0\,\vec{V}_7$	$0.0\,\vec{V}_7$	$0.0\,\vec{V}_7$	$0.0\,\vec{V}_7$	$0.0\,\vec{V}_7$	$0.0\,\vec{V}_7$	$1.0\,\vec{V}_7$	$1.0\,\vec{V}_7$
	+	+	+	+	+	+	+	+
$\vec{E}_8 \xrightarrow{W_v} \vec{V}_8$	$0.0\,\vec{V}_8$	$0.0\,\vec{V}_8$	$0.0\,\vec{V}_8$	$0.0\,\vec{V}_8$	$0.0\,\vec{V}_8$	$0.0\,\vec{V}_8$	$0.0\,\vec{V}_8$	$0.0\,\vec{V}_8$
	$\Vert$	$\Vert$	$\Vert$	$\Vert$	$\Vert$	$\Vert$	$\Vert$	$\Vert$
	$\Delta\vec{E}_1$	$\Delta\vec{E}_2$	$\Delta\vec{E}_3$	$\Delta\vec{E}_4$	$\Delta\vec{E}_5$	$\Delta\vec{E}_6$	$\Delta\vec{E}_7$	$\Delta\vec{E}_8$

EURO AUSTRIA

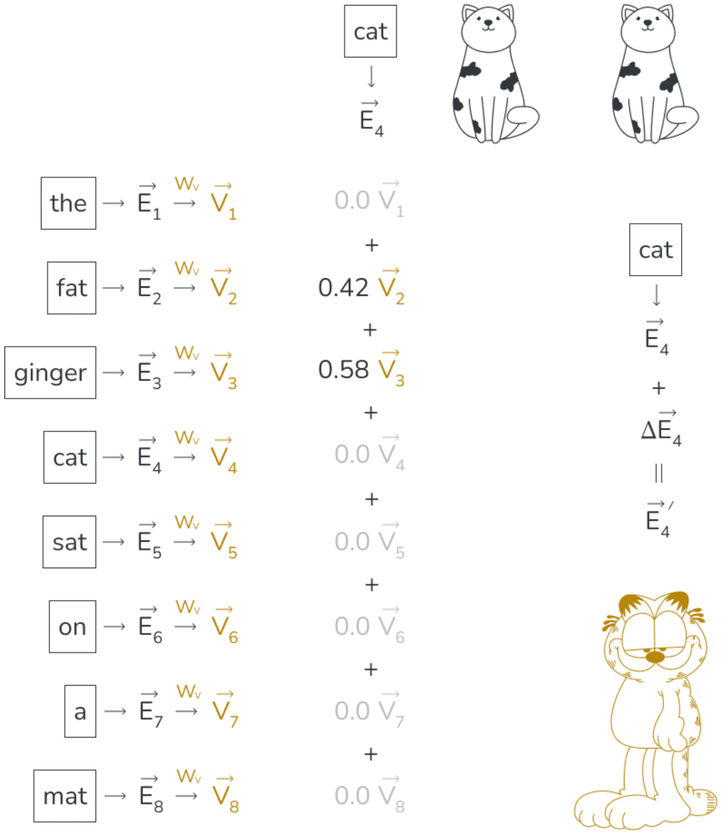
Attention! Queries, Keys and Values

Value

Eeach value vector is then multiplied by the corresponding weight for this word and the results summed up.

The result $\Delta\vec{E_i}$ is the change, that needs to be added to the origial embedding to get an updated, richer meaning.

Since this happens to every word in the sequence, we end up with a set of more refined embeddings.



Attention

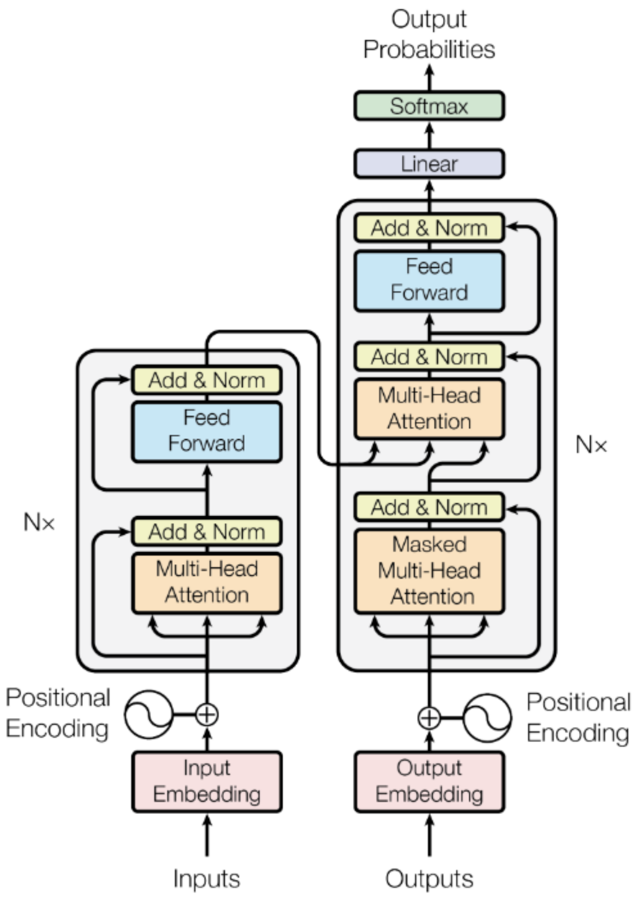
Multi-Head Attention

The attention mechanism we just looked at, is done several times in parallel.

Each attention head is focussing on different features of the embeddings and computes ist own $\Delta\overrightarrow{E_i}^{(j)}$

The resulting $\Delta\overrightarrow{E_i}^{(j)}$ vectors, each suggesting the necessary change, are added to the orginal embedding.

This can be done in parallel on GPUs.



Source: “Attention Is All You Need”, Vaswani et al.

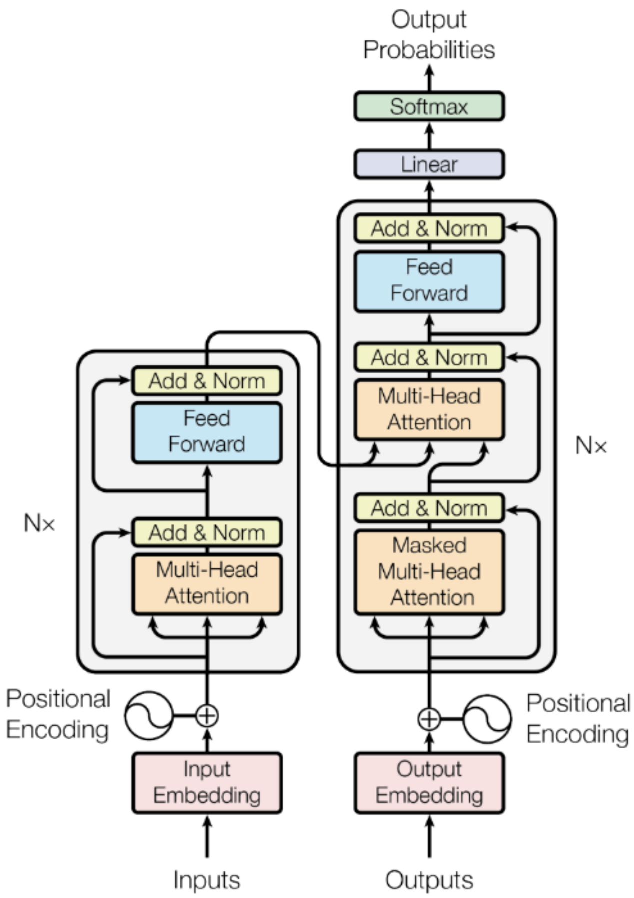
Attention

Cross Attention

Cross attention is almost the same as self attention.

Difference:

- query and key maps act on different data sets (e.g. 2 different languages in machine translation)
- no masking, since there is no issue of later words affecting earlier ones.

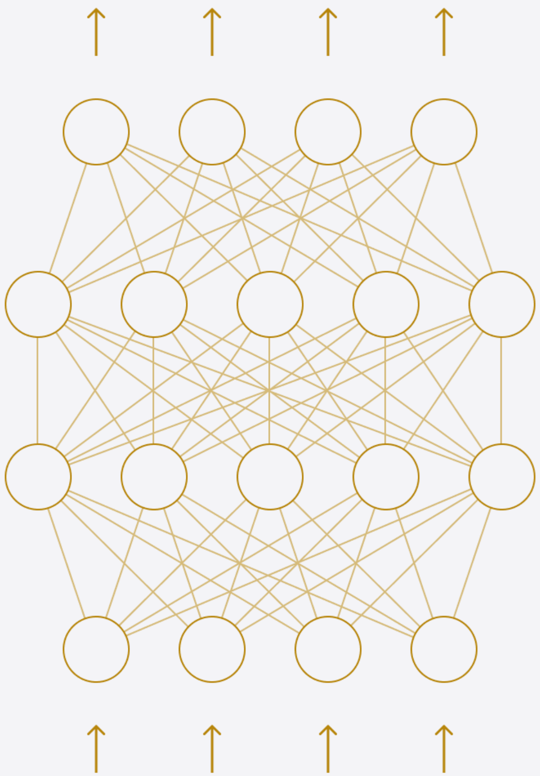


Source: “Attention Is All You Need”, Vaswani et al.

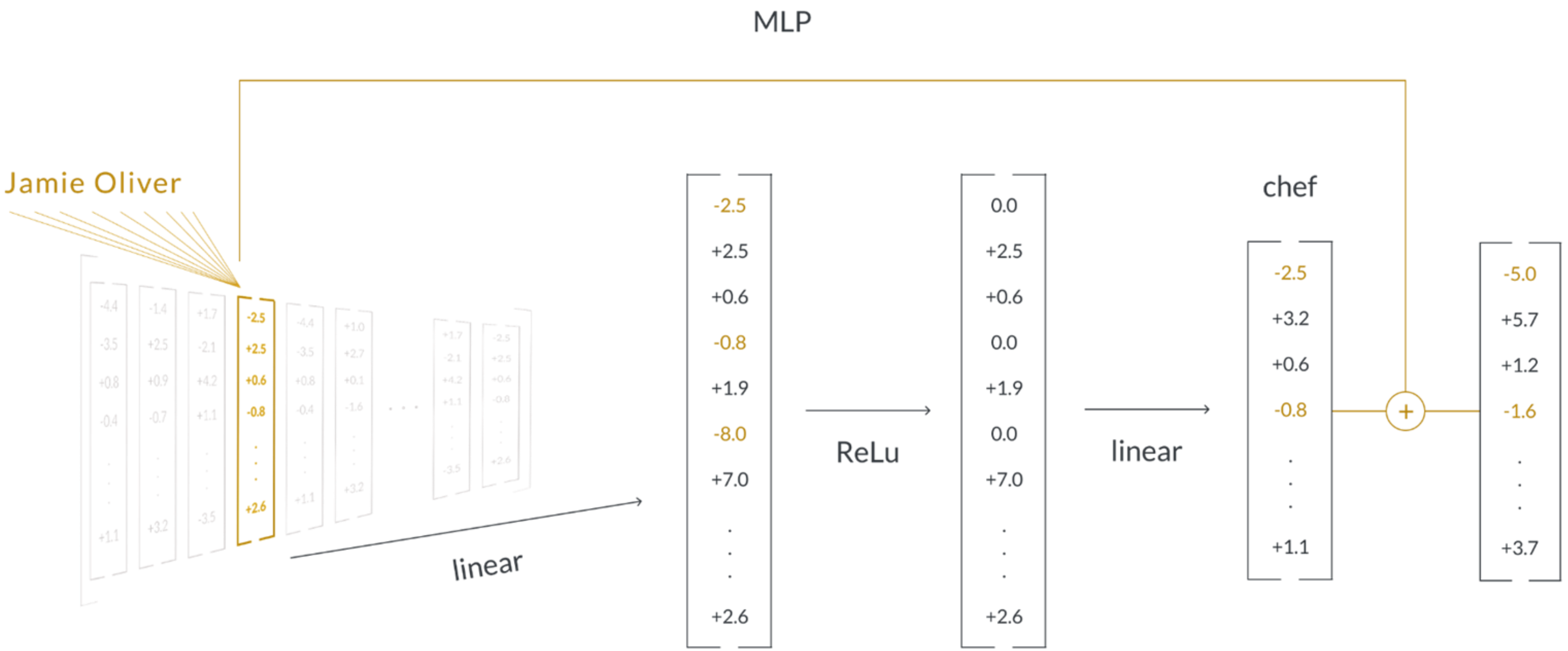
Feed Forward Network

Multi Layer Perceptron (MLP)

- Home to approx. 2/3s of all parameters
- This is where „knowledge“ is baked in
- Source of halluzinations
- Facts that are associated with input embeddings are added to the input embeddings
- Each embedding vector can be processed independently -> parallelization!



Feed Forward Network



THANK YOU

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101101903. The JU receives support from the Digital Europe Programme and Germany, Bulgaria, Austria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia