CS540 Summer 2023

Attention and Transformers

Jiang, Yuye

PhD student in Computer Sciences

Participation game (on TopHat)

What is the full name of "GPT"?

A. Generic Pre-trained Transformers
B. Generic Pre-trained Tensors
C. Generative Pre-trained Transformers
D. Generative Probabilistic Transformer

Background

- Attention Mechanism
  - Bahdanau et. al. 2014. Neural Machine Translation by Jointly Learning to Align and Translate.
  - Originally developed as an enhancement of RNN applied to translation task
- Transformer Model
  - Vaswani et. al. 2017. Attention is All you Need.
  - First transduction model relying entirely on self-attention to compute representations of its input and output
  - Backbone of the modern large language models & CV models, etc
  - GPT = Generative Pre-trained **Transformer**

Database Manager

The thing your customer is interested in

Query
2

Database of (Key, Value) pairs

Key	Value
1	100
2	200
3	300

Database Manager

The thing your customer is interested in

Query
2

Database of (Key, Value) pairs

Key	Value
1	100
2	200
3	300

unrelated to 2
equal to 2
unrelated to 2

Database Manager

The thing your customer is interested in

Query
2

Database of (Key, Value) pairs

Key	Value		Attention weight	
1	100	unrelated to 2	0	$0\% \times 100$
2	200	equal to 2	1	$100\% \times 200$
3	300	unrelated to 2	0	$0\% \times 300$

Output
200

Database Manager

The thing your customer is interested in

Query
2.5

Database of (Key, Value) pairs

Key	Value		Attention weight	
1	100	far from 2.5	0	$0\% \times 100$
2	200	close to 2.5	0.5	$50\% \times 200$
3	300	close to 2.5	0.5	$50\% \times 300$

Output
250

Attention Mechanism

$$Attention(q, D) = \sum_{i=1}^{m} \alpha(q, k_i)v_i$$

q

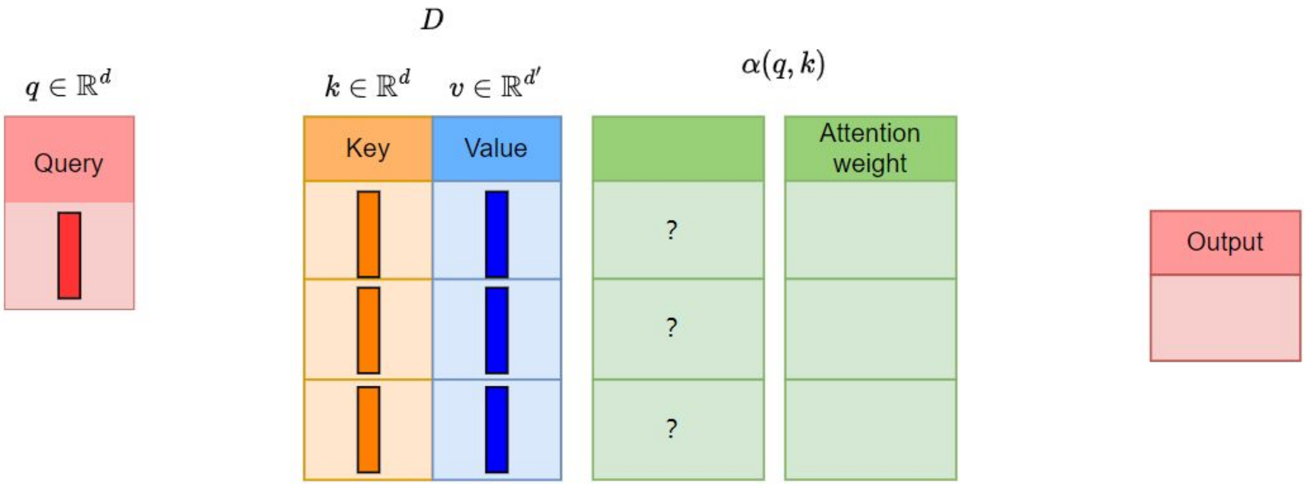
Query
2

D		$\alpha(q,k)$		$\sum_{i=1}^{m} \alpha(q, k_i)v_i$
Key	Value		Attention weight	
1	100	unrelated to 2	0	$0\% \times 100$
2	200	equal to 2	1	$100\% \times 200$
3	300	unrelated to 2	0	$0\% \times 300$

Output
200

Attention Mechanism

$$Attention(q, D) = \sum_{i=1}^{m} \alpha(q, k_i) v_i$$

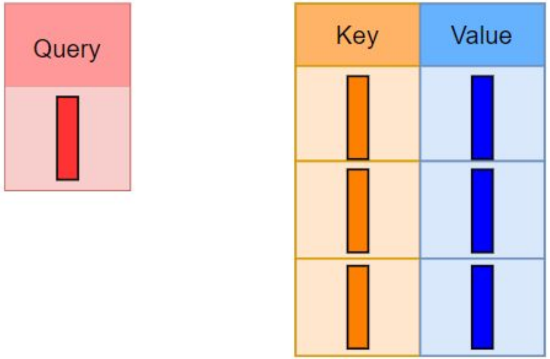


Attention Mechanism

$$Attention(q, D) = \sum_{i=1}^{m} \alpha(q, k_i) v_i$$

D

$q \in \mathbb{R}^d$ $k \in \mathbb{R}^d$ $v \in \mathbb{R}^{d'}$



$\alpha(q, k)$

Attention Scoring function $a(q, k)$

Any function that captures similarity between q and k for your task

A common one: scaled dot-product attention

$$a(q, k) = \frac{q^T k}{\sqrt{d}}$$

vector direction more similar → dot-product higher
See: cosine similarity

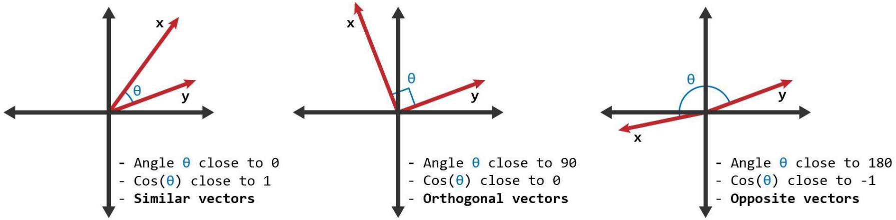
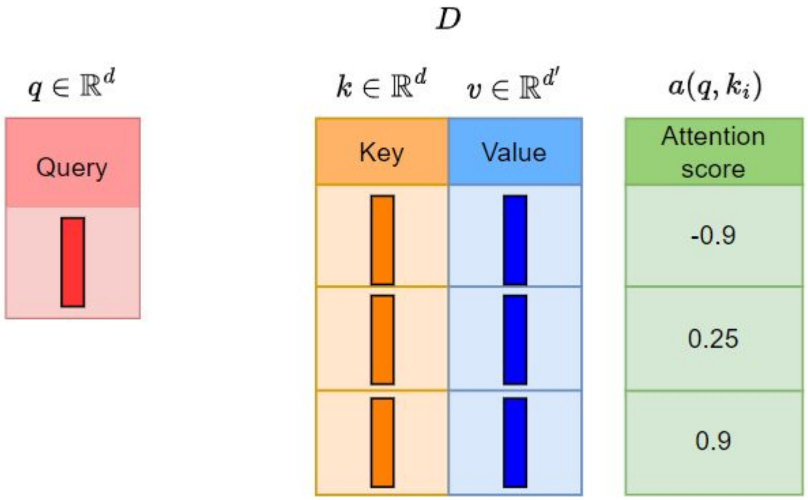


image source

Attention Mechanism

$$Attention(q, D) = \sum_{i=1}^{m} \alpha(q, k_i) v_i$$



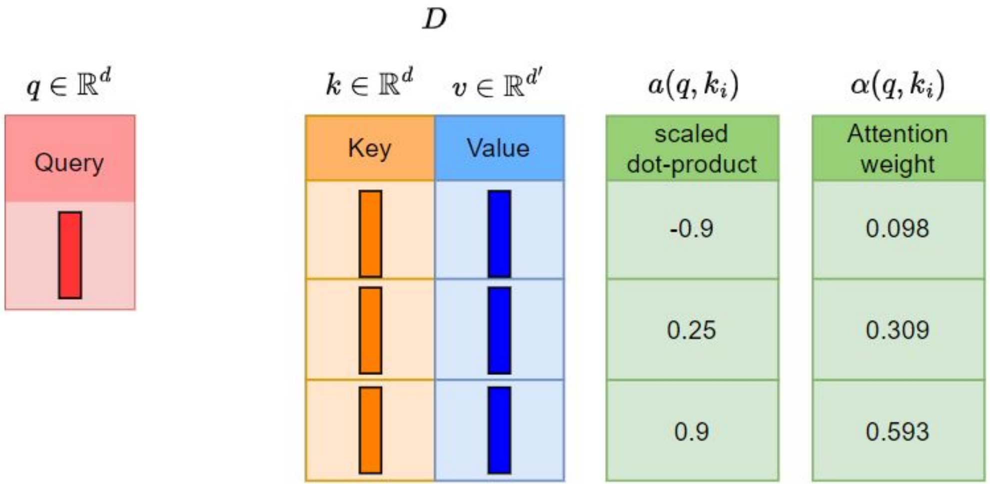
Attention score does not follow probability distribution

convert this to a probability via softmax function

$$softmax(a(q, k_i)) = \frac{e^{a(q,k_i)}}{\sum_{j=1}^{m} e^{a(q,k_j)}}$$

Attention Mechanism

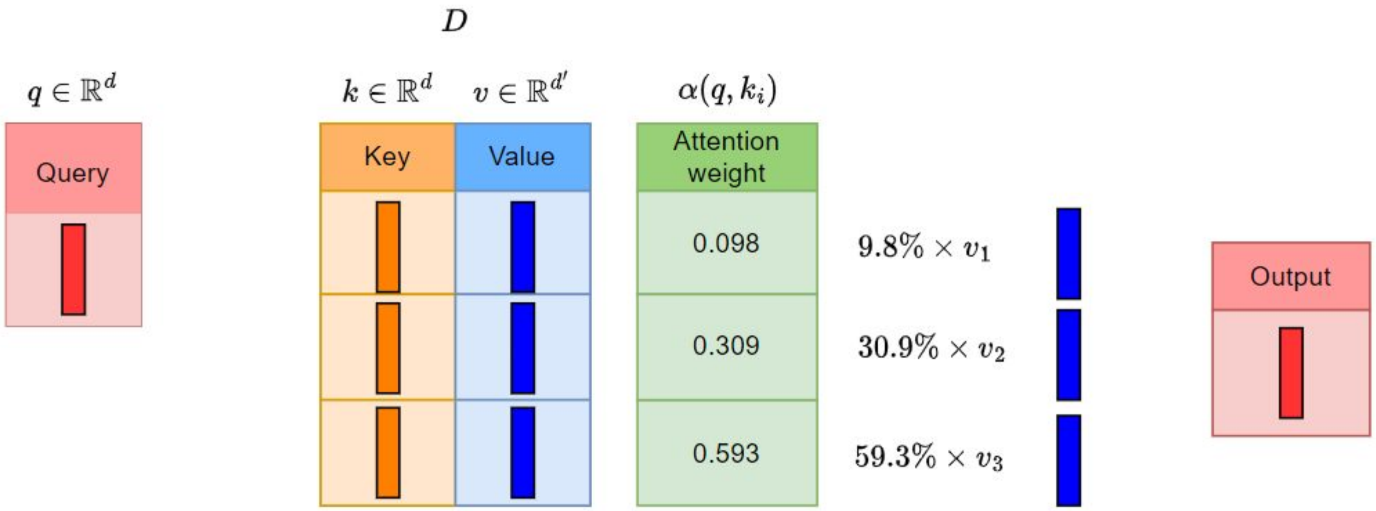
$$Attention(q, D) = \sum_{i=1}^{m} \alpha(q, k_i) v_i$$



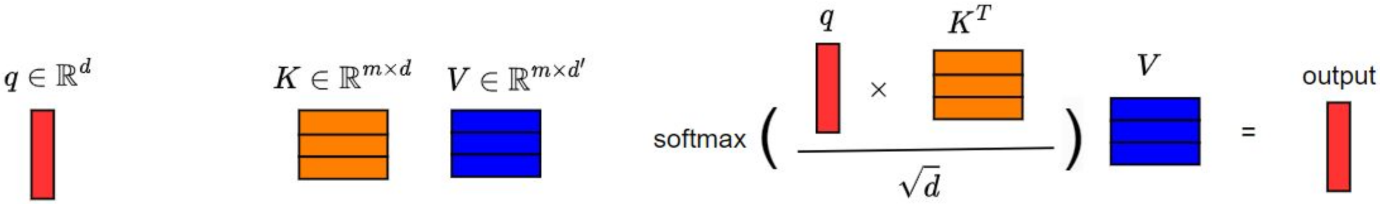
$$\alpha(q, k_i) = softmax(a(q, k_i))$$

Attention Mechanism

$$Attention(q, D) = \sum_{i=1}^{m} \alpha(q, k_i) v_i$$

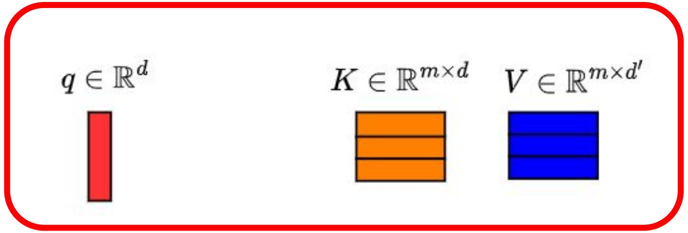


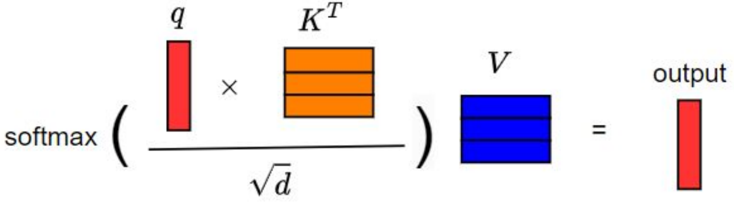
Attention Mechanism



$$Attention(q, K, V) = softmax(\frac{qK^T}{\sqrt{d}})V$$

Attention Mechanism

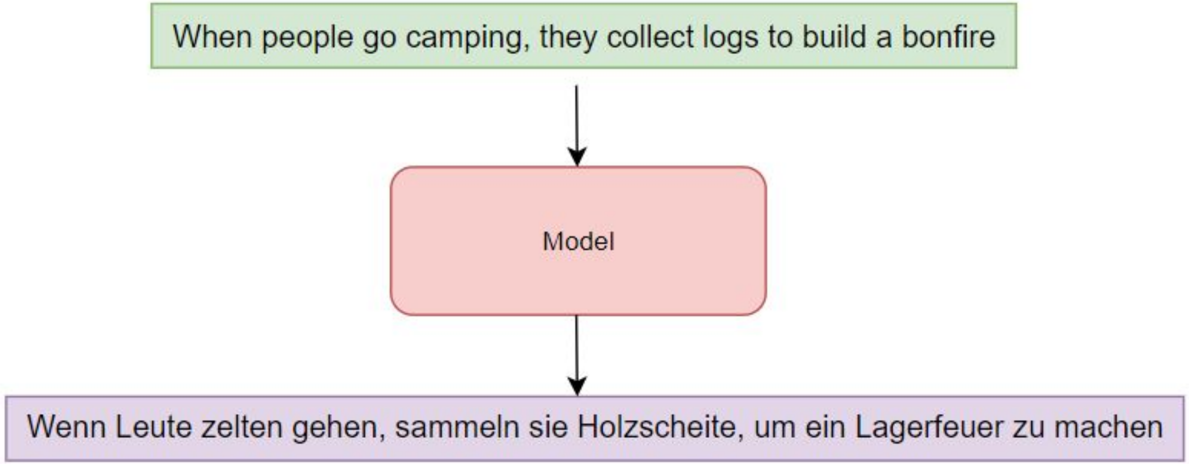




But how are these constructed?

$$Attention(q, K, V) = softmax(\frac{qK^T}{\sqrt{d}})V$$

Machine Translation



Embedding

A cat is sleeping on a red sofa

A dog is sitting on a green chair

one-hot encoding:

vocab		cat =	dog =
	a	0	0
	cat	1	0
	is	0	0
	sleeping	0	0
	on	0	0
	red	0	0
	sofa	0	0
	dog	0	1
	sitting	0	0
	green	0	0
	chair	0	0

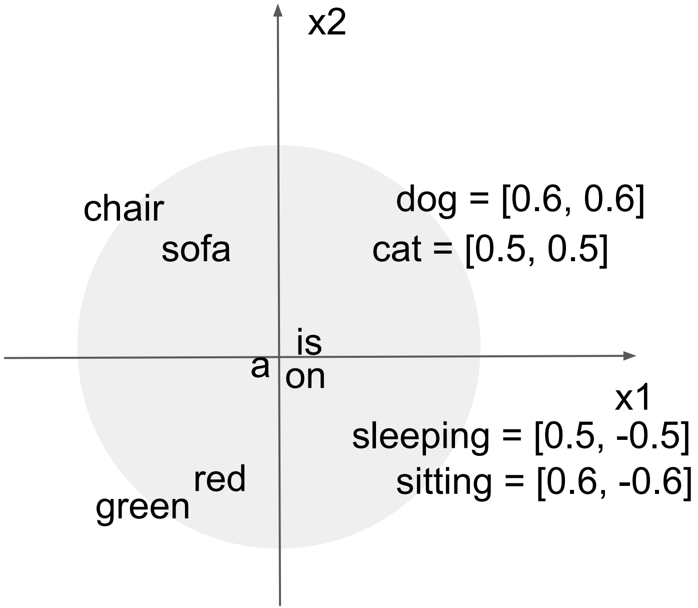
Can be used for numerical computation

No similarity measurement

cannot tell “cat” & “dog” similar
and tend to appear in similar context & position

Embedding

2-d embedding



A cat is sleeping on a red sofa

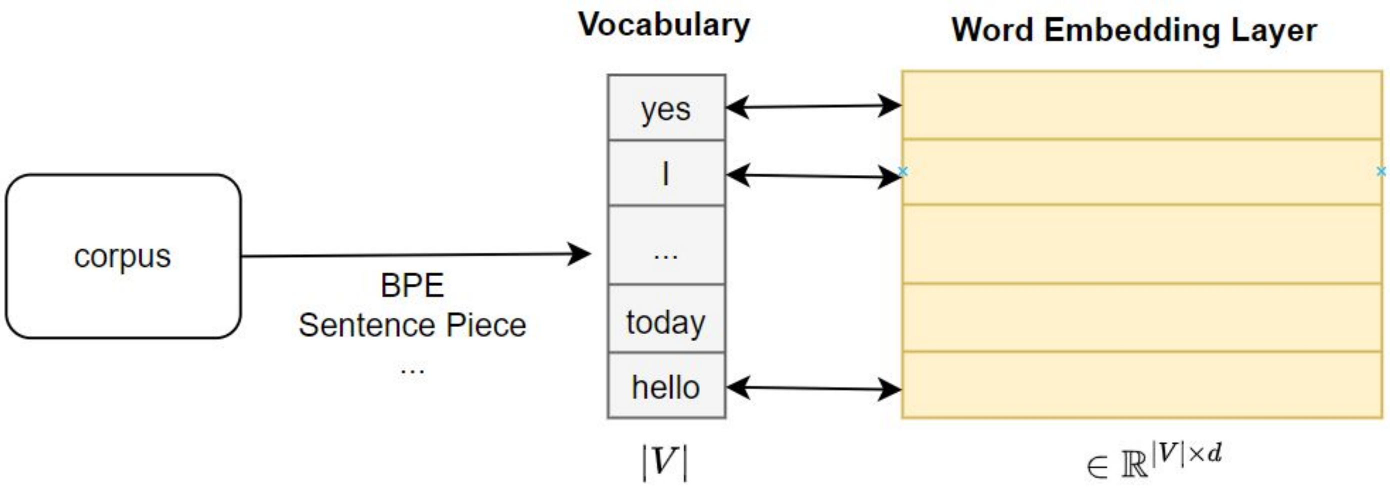
A dog is sitting on a green chair

Each word is converted to a vector

- can input to neural network
- can learn similarity between words

For the many words in languages, usually pick a very large embedding dimension, for example d = 768

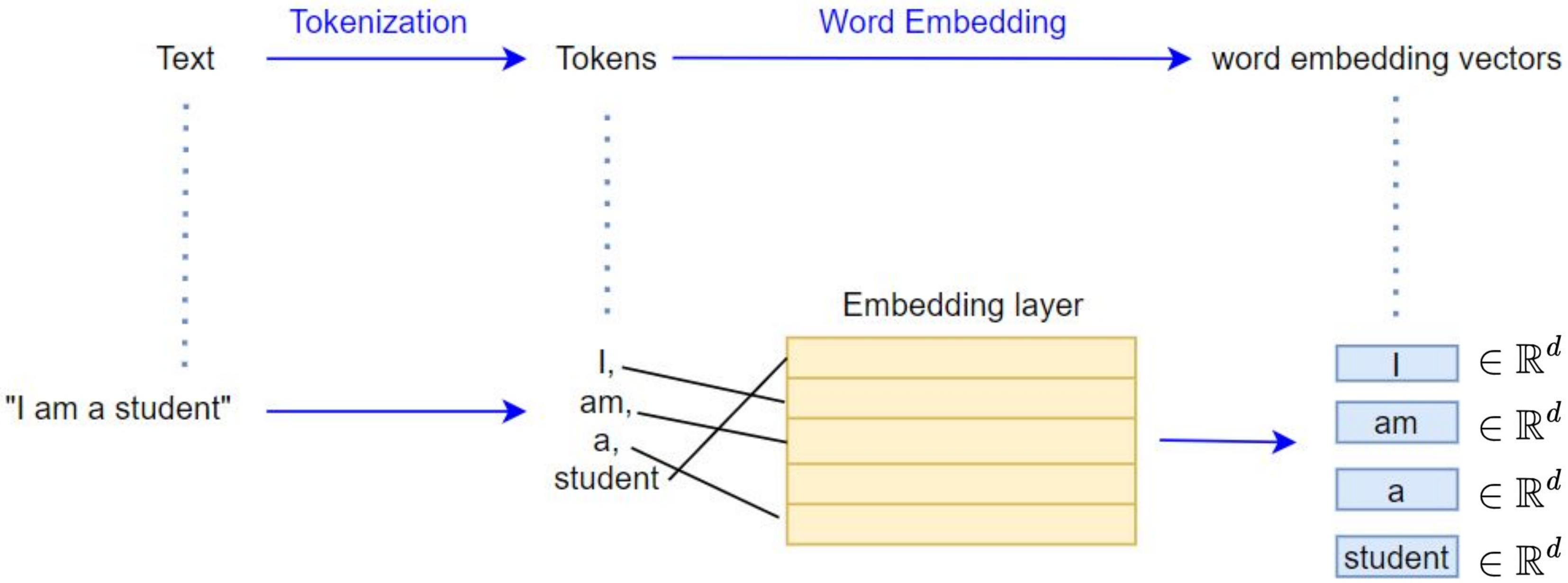
Embedding



Learnable and updated with the model

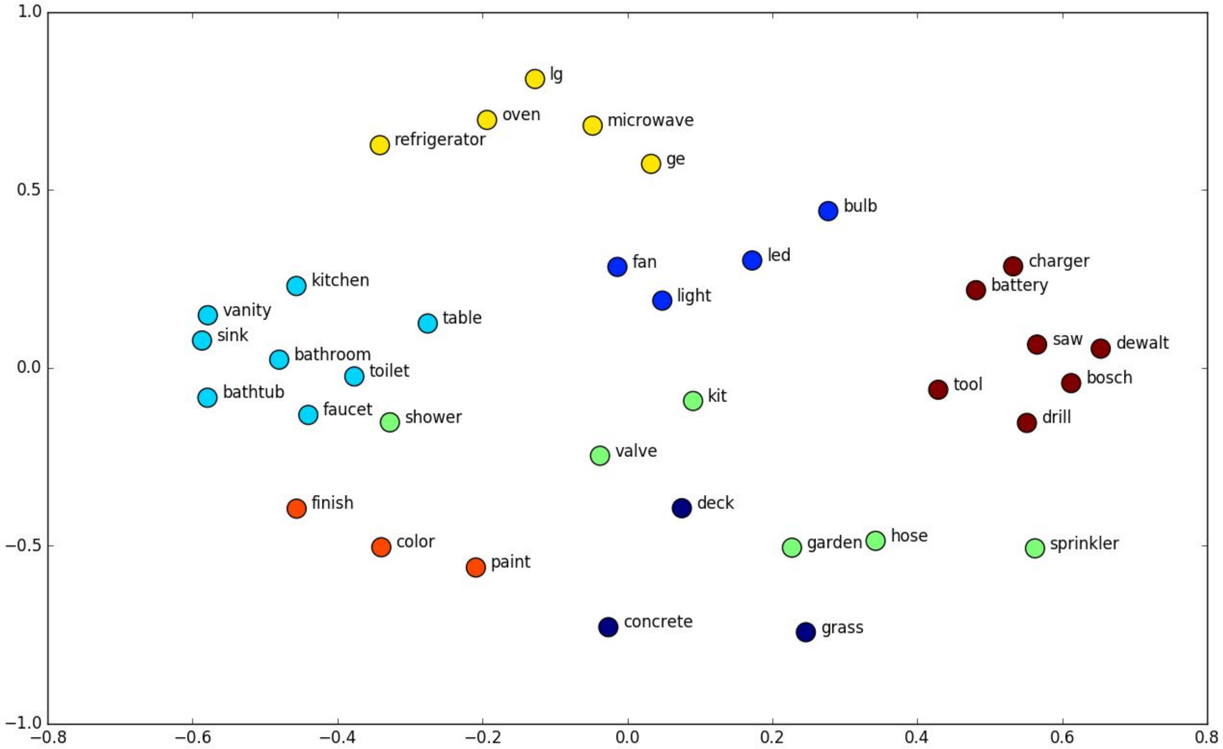
d: embedding dimension, usually picked to be a high number like 768

Embedding



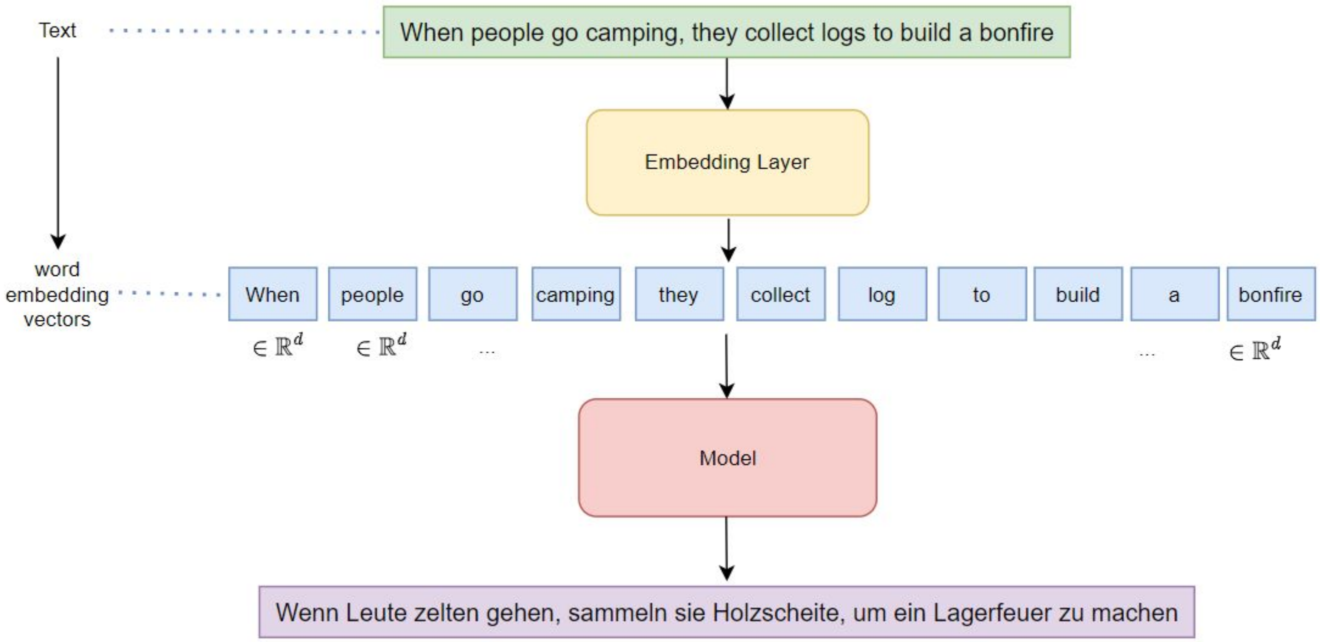
Tokenization
Word Embedding
Text
Tokens
word embedding vectors
Embedding layer
"I am a student"
I,
am,
a,
student
I
am
a
student
$\in \mathbb{R}^d$
$\in \mathbb{R}^d$
$\in \mathbb{R}^d$
$\in \mathbb{R}^d$

Embedding



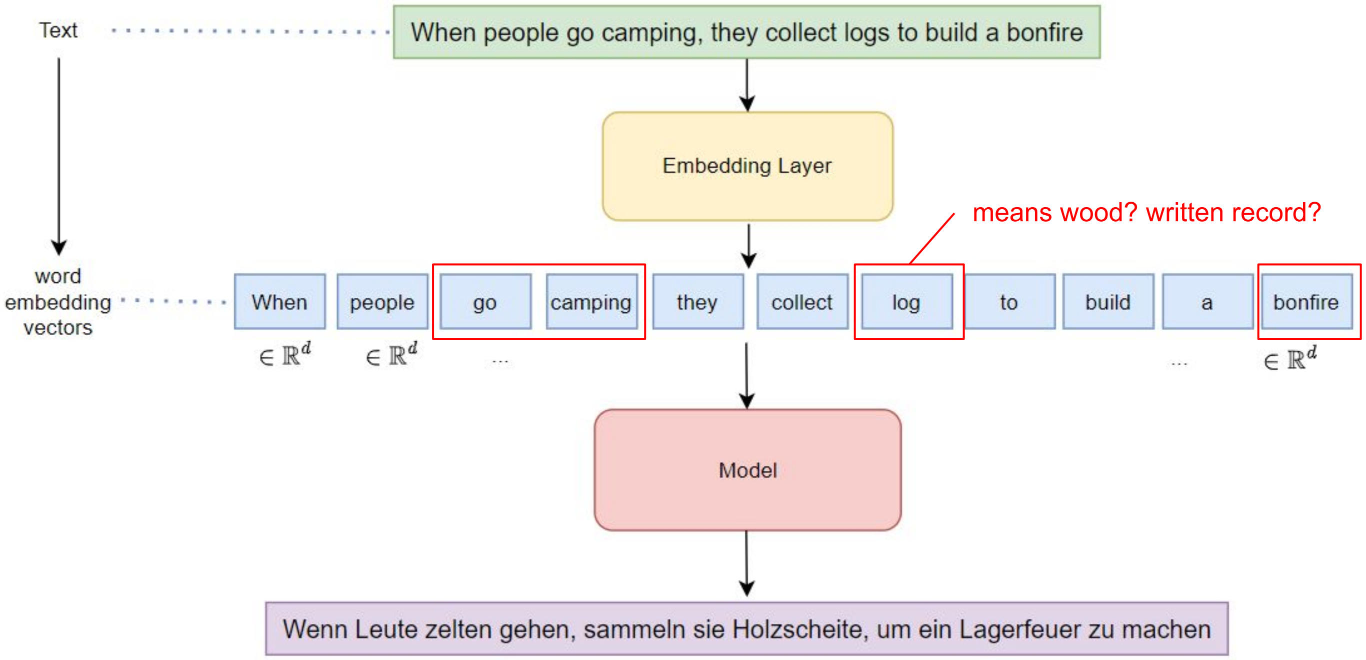
[image source](#)

Embedding



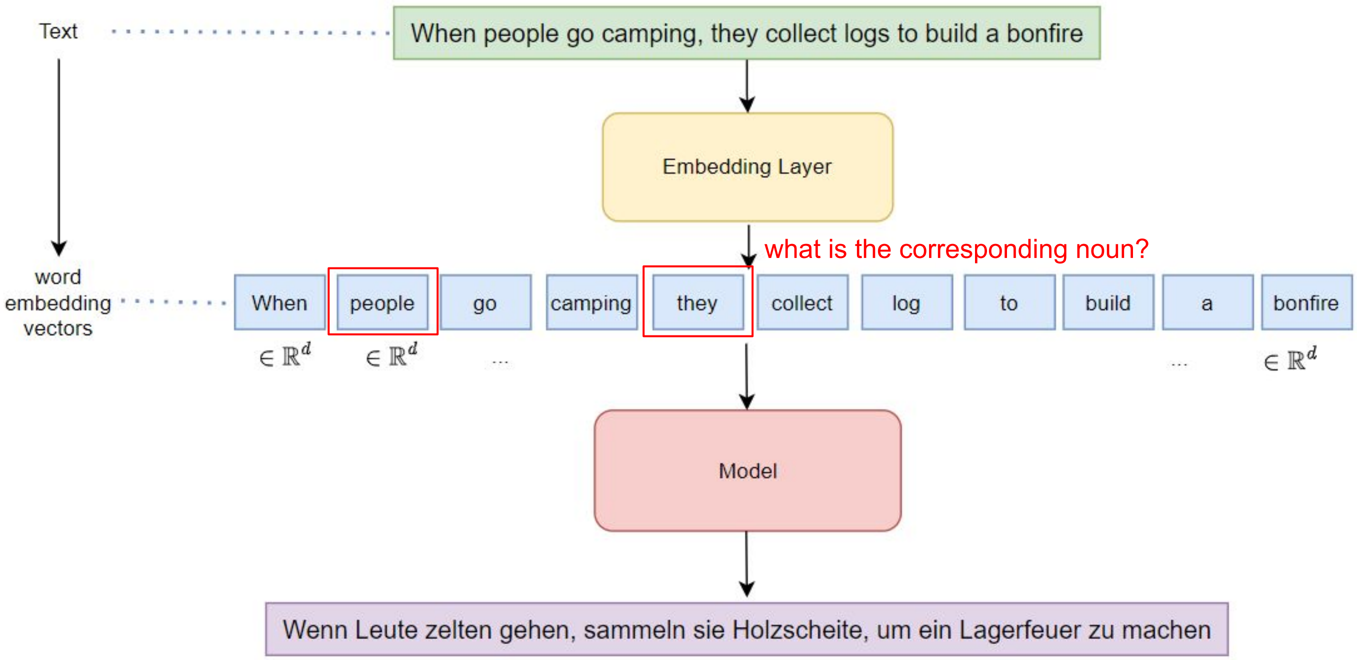
Text
When people go camping, they collect logs to build a bonfire
Embedding Layer
word embedding vectors
When
people
go
camping
they
collect
log
to
build
a
bonfire
$\in \mathbb{R}^d$
$\in \mathbb{R}^d$
...
...
$\in \mathbb{R}^d$
Model
Wenn Leute zelten gehen, sammeln sie Holzscheite, um ein Lagerfeuer zu machen

Embedding



Text
When people go camping, they collect logs to build a bonfire
Embedding Layer
means wood? written record?
word embedding vectors
When
people
go
camping
they
collect
log
to
build
a
bonfire
$\in \mathbb{R}^d$
$\in \mathbb{R}^d$
...
...
$\in \mathbb{R}^d$
Model
Wenn Leute zelten gehen, sammeln sie Holzscheite, um ein Lagerfeuer zu machen

Embedding



Text
When people go camping, they collect logs to build a bonfire
Embedding Layer
what is the corresponding noun?
word embedding vectors
When
people
go
camping
they
collect
log
to
build
a
bonfire
$\in \mathbb{R}^d$
$\in \mathbb{R}^d$
...
...
$\in \mathbb{R}^d$
Model
Wenn Leute zelten gehen, sammeln sie Holzscheite, um ein Lagerfeuer zu machen

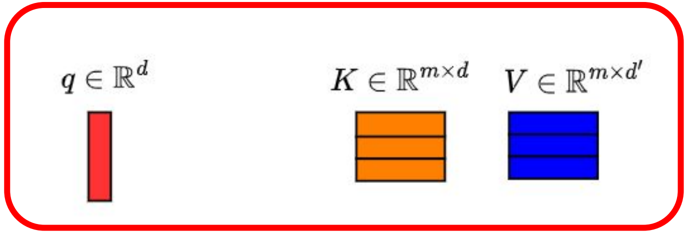
Self-attention

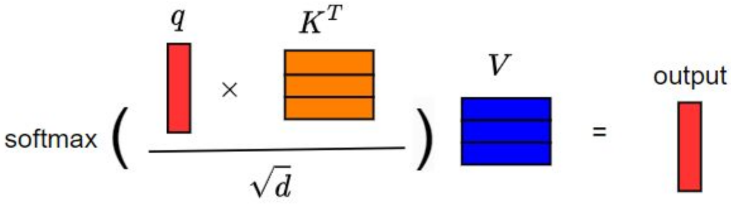
When | people | go | camping | they | collect | log | to | build | a | bonfire

“Every word needs to pay attention to each other”

“Every word should pay more attention to the other word thats is related to it”

Reminder of our initial question

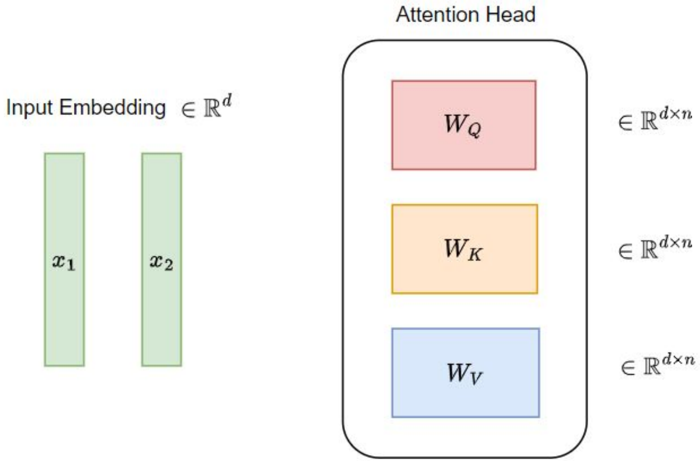




But how are these constructed?

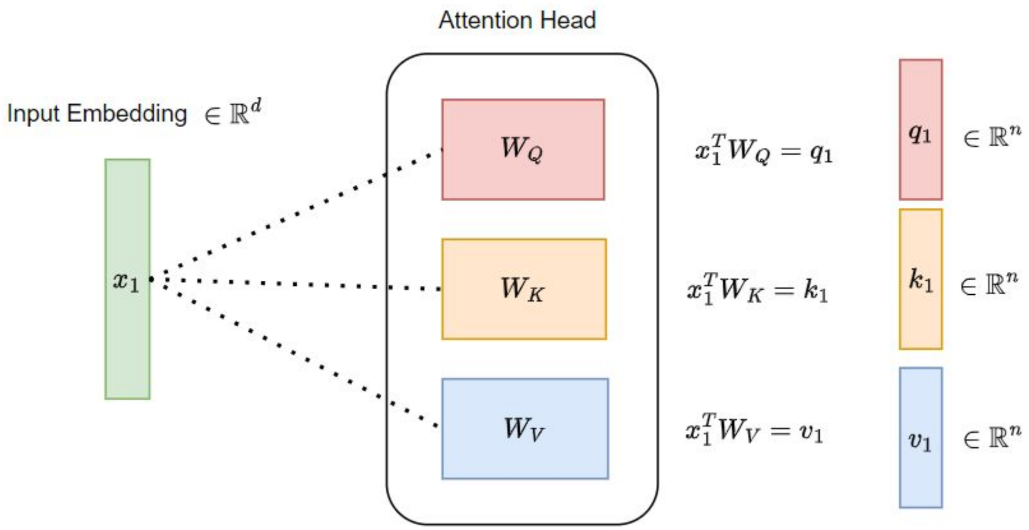
$$Attention(q, K, V) = softmax(\frac{qK^T}{\sqrt{d}})V$$

Attention Head



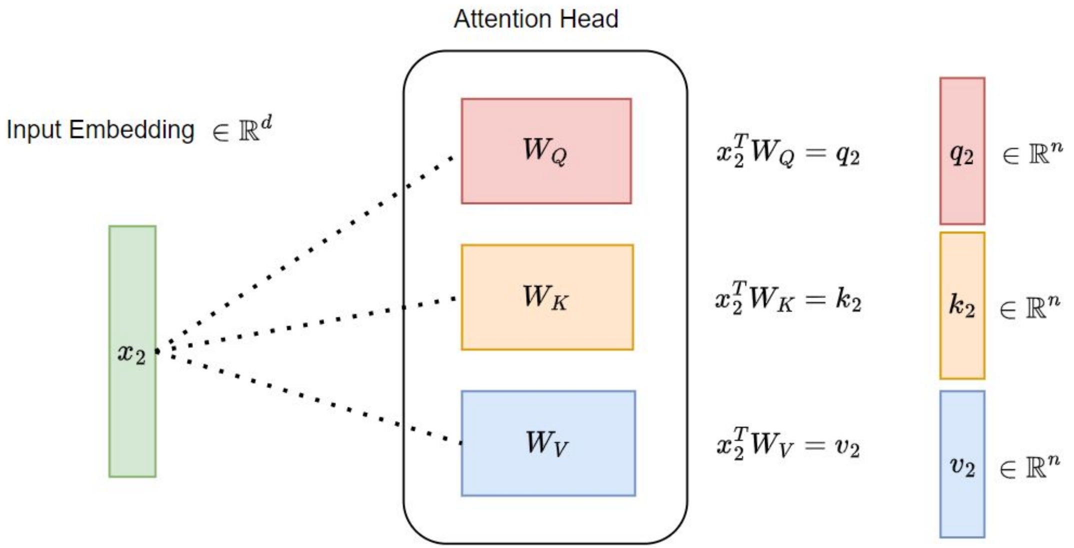
Attention Head

Multiply the first embedding vector with each matrix

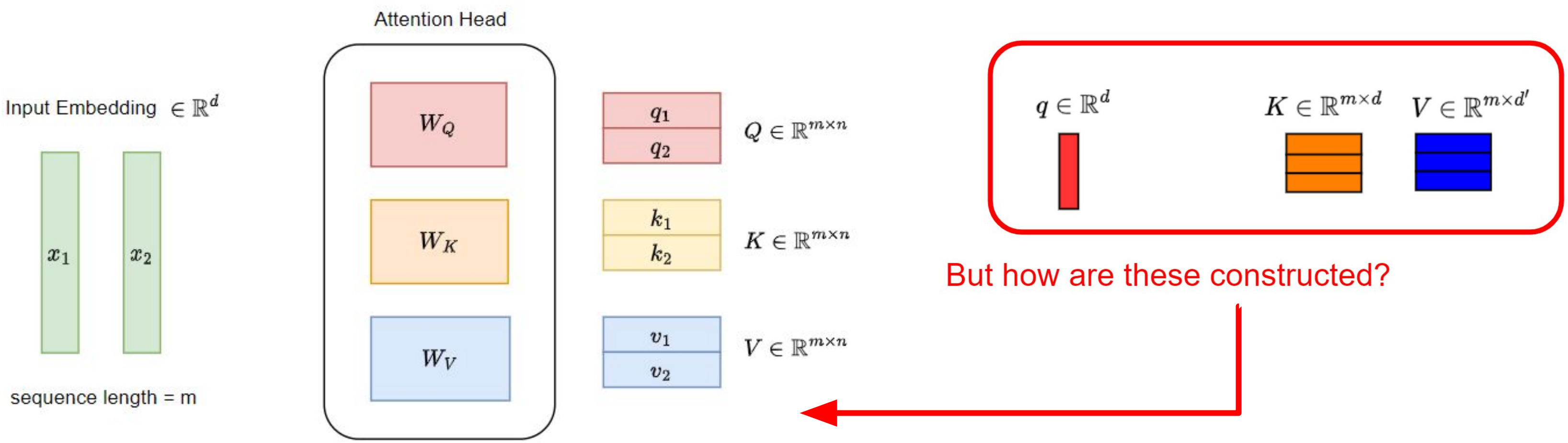


Attention Head

Multiply the second embedding vector with each matrix



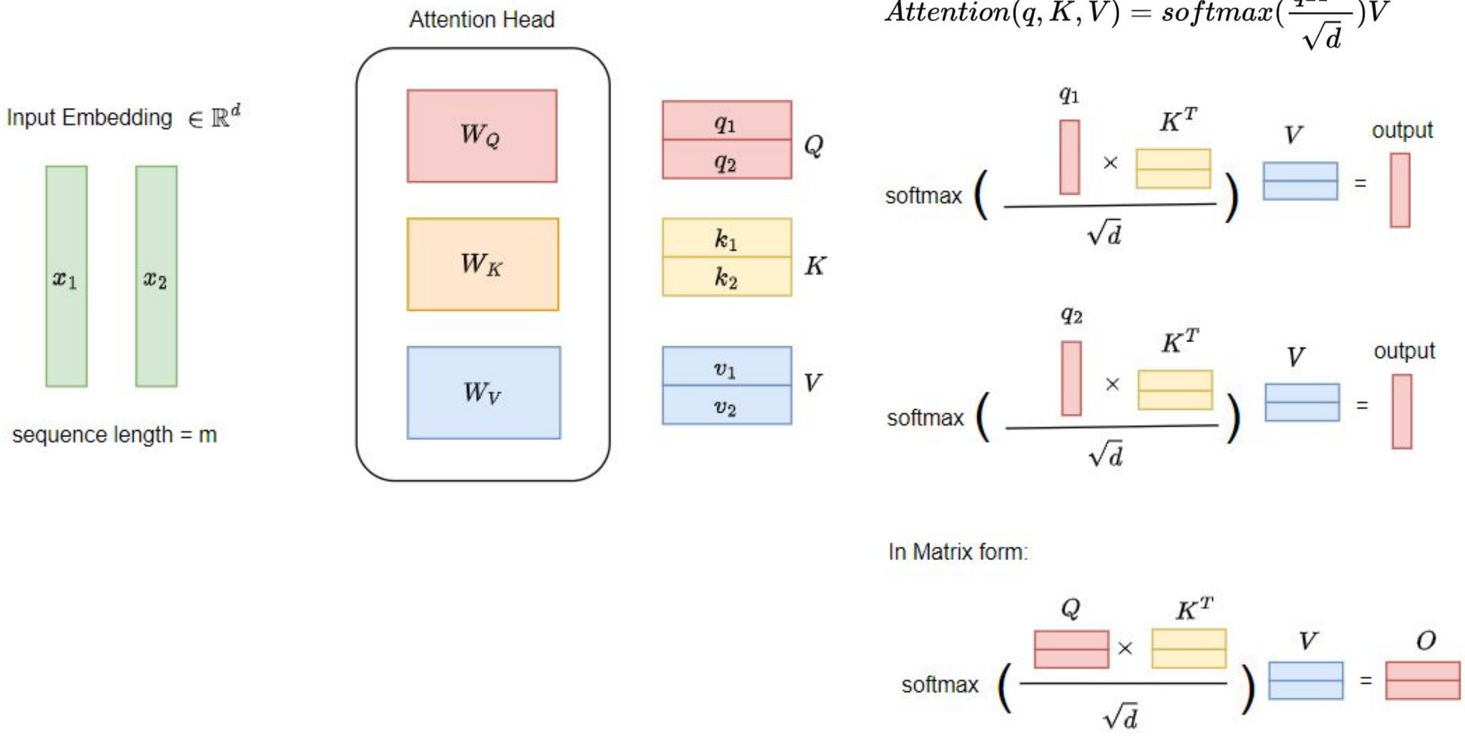
Attention Head



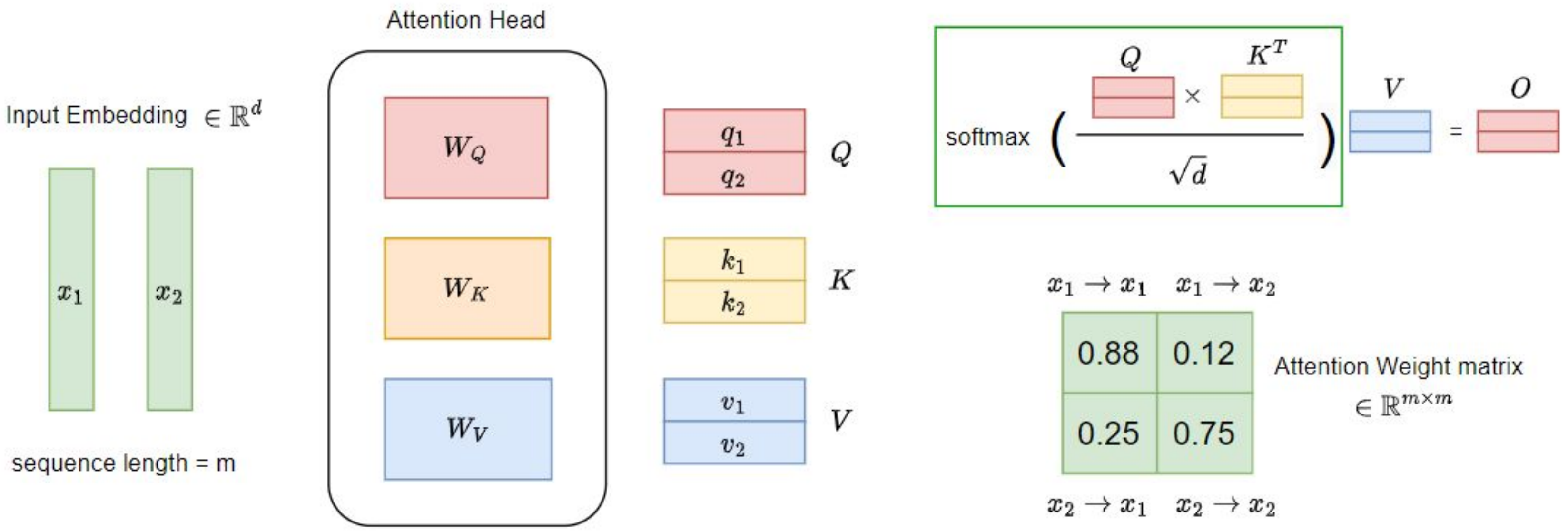
The query, key and value for each word is calculated from all the m words in the same sentence, using shared learnable matrices

Attention Head

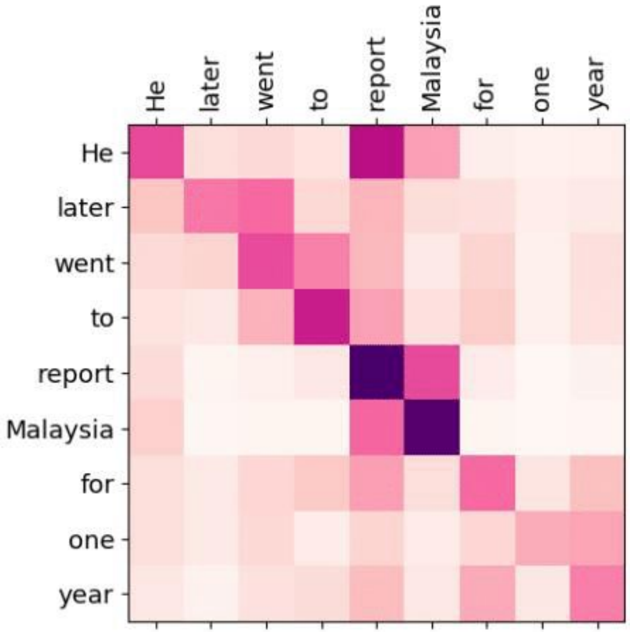
$$Attention(q, K, V) = softmax(\frac{qK^T}{\sqrt{d}})V$$



Attention Head



Attention Head



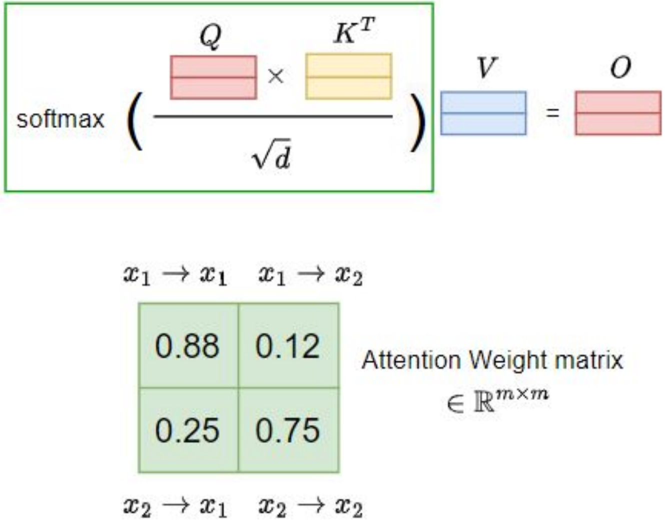
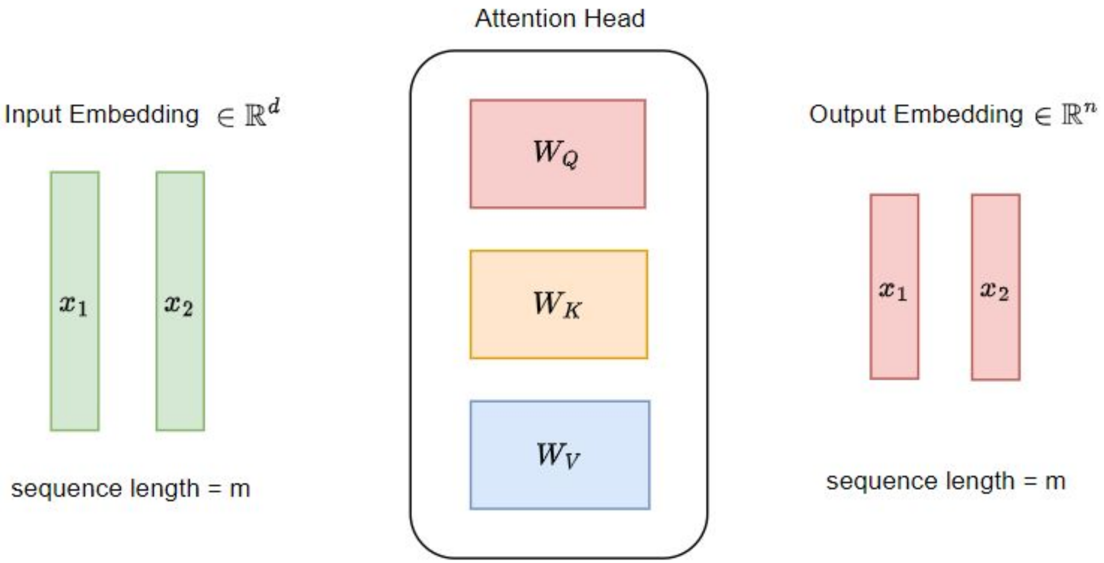


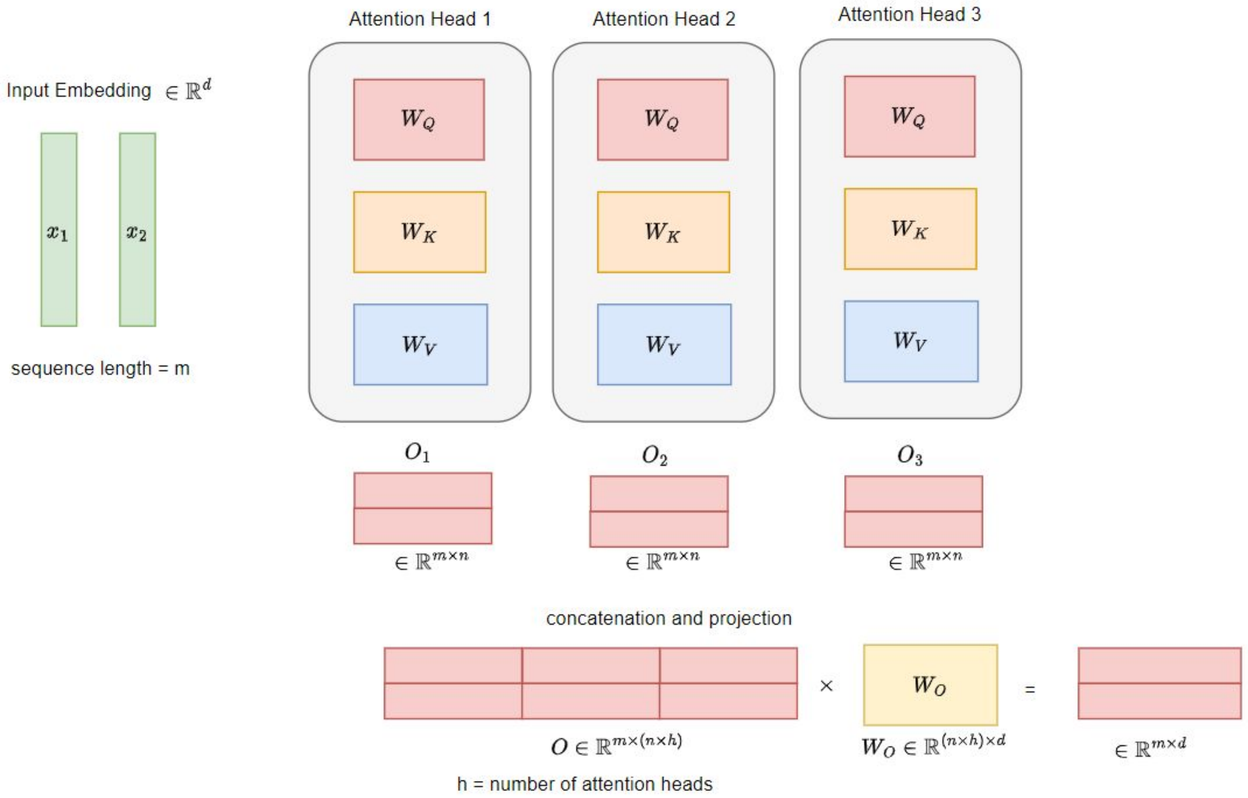
image source

Attention Head

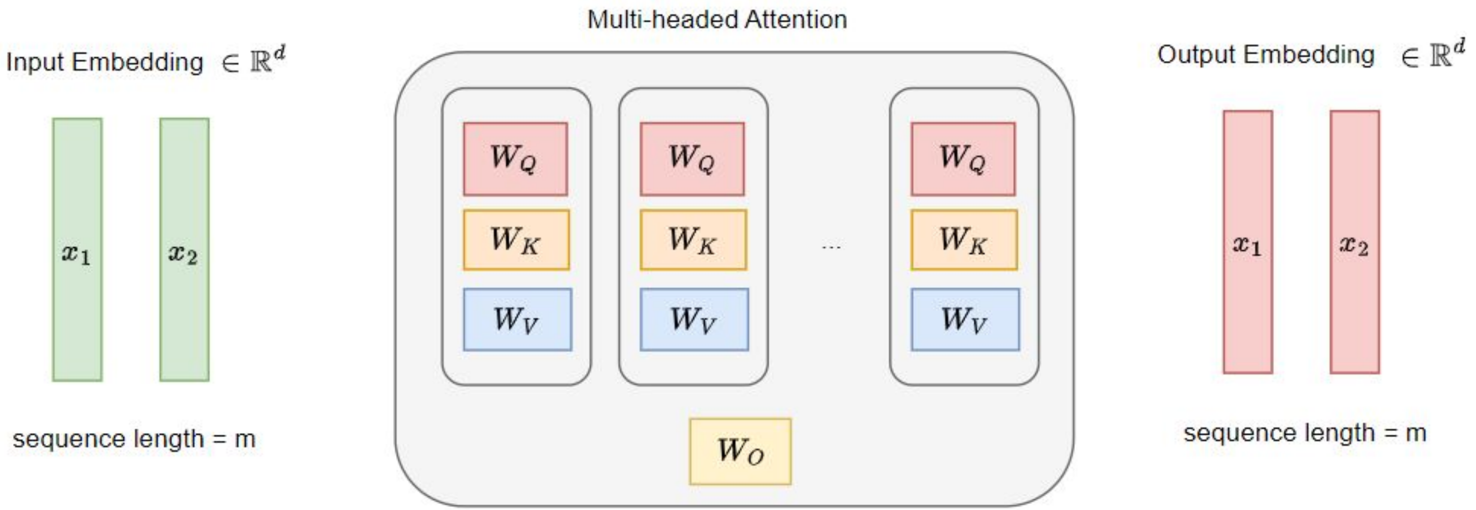


learnable parameters

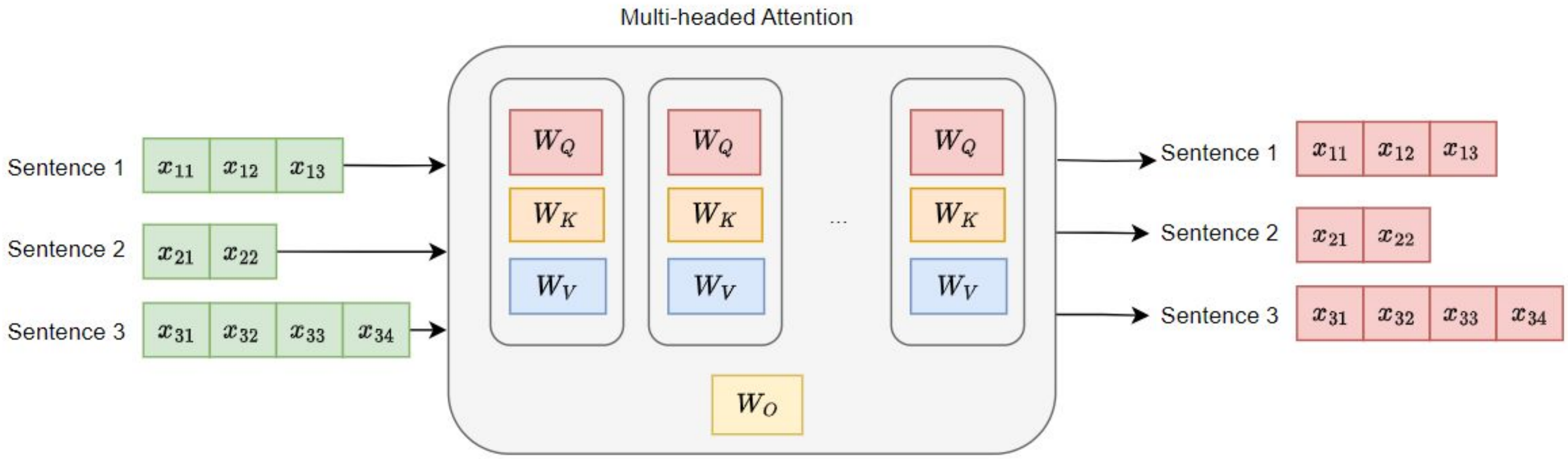
Multi-headed Attention



Multi-headed Attention



Multi-headed Attention



Batch of input sentences

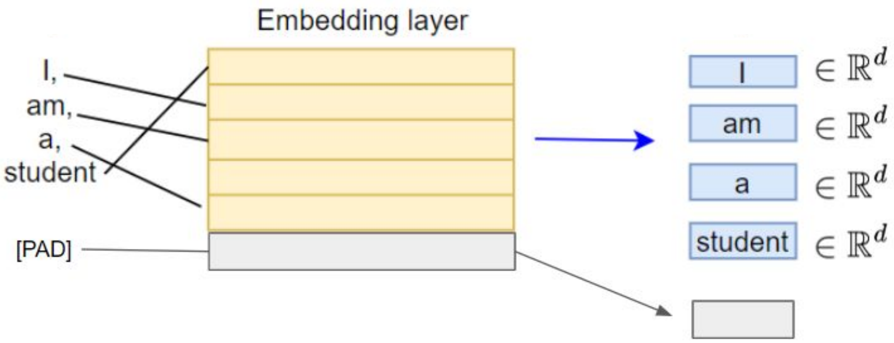
Good morning
How are you
Hello

Good morning [PAD]
How are you
Hello [PAD] [PAD]

Batch = 3

Good	morning	[PAD]
How	are	you
Hello	[PAD]	[PAD]

$\in \mathbb{R}^{b \times maxlen \times d}$



Insert padding

maxlen = maximum sequence length in this batch

Don't want padding to affect training

Attention Mask

Good	morning	[PAD]
How	are	you
Hello	[PAD]	[PAD]

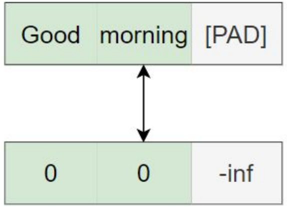
input

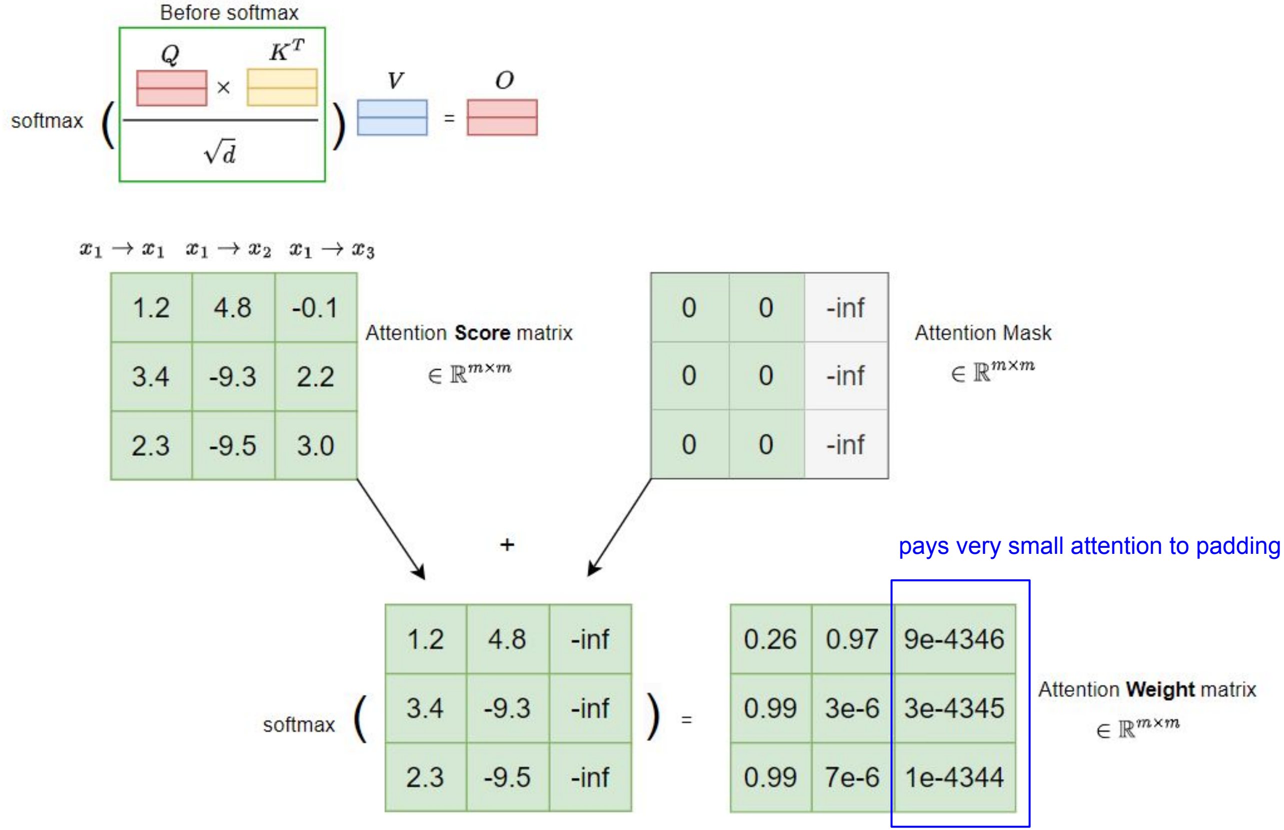
0	0	-inf
0	0	0
0	-inf	-inf

attention mask

-inf represents a very small negative number

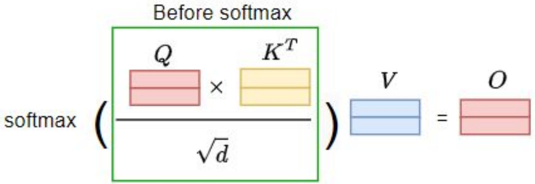
Attention Mask





Attention Mask

Good	morning	[PAD]
0	0	-inf



$x_1 \to x_1$	$x_1 \to x_2$	$x_1 \to x_3$
1.2	4.8	-0.1
3.4	-9.3	2.2
2.3	-9.5	3.0

Attention **Score** matrix $\in \mathbb{R}^{m \times m}$

+

0	0	-inf
0	0	-inf
0	0	-inf

Attention Mask $\in \mathbb{R}^{m \times m}$

softmax (

1.2	4.8	-inf
3.4	-9.3	-inf
2.3	-9.5	-inf

) =

0.26	0.97	9e-4346
0.99	3e-6	3e-4345
0.99	7e-6	1e-4344

Attention **Weight** matrix $\in \mathbb{R}^{m \times m}$

what about pad's attention to other tokens?

Batch of input sentences

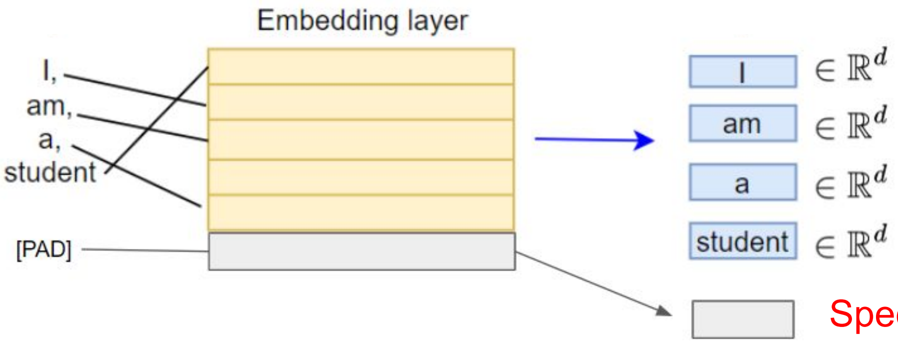
Good morning
How are you
Hello

Good morning [PAD]
How are you
Hello [PAD] [PAD]

Batch = 3

Good	morning	[PAD]
How	are	you
Hello	[PAD]	[PAD]

$\in \mathbb{R}^{b \times maxlen \times d}$



Special vector
The gradient is not calculated

Attention Mask in libraries

input

Good	morning	[PAD]
How	are	you
Hello	[PAD]	[PAD]

attention mask

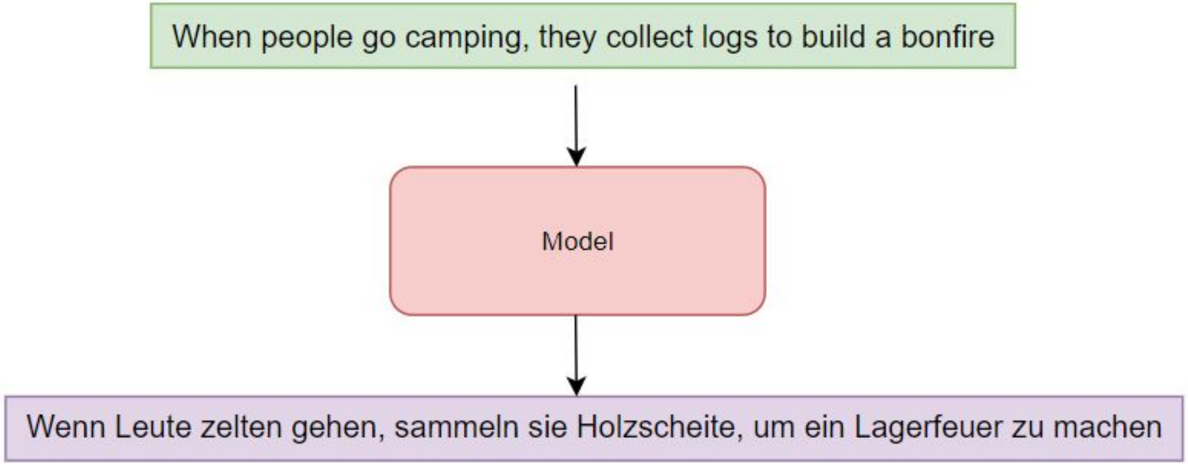
1	1	0
1	1	1
1	0	0

implementation from
pytorch
huggingface
…

do the transformation inside for you

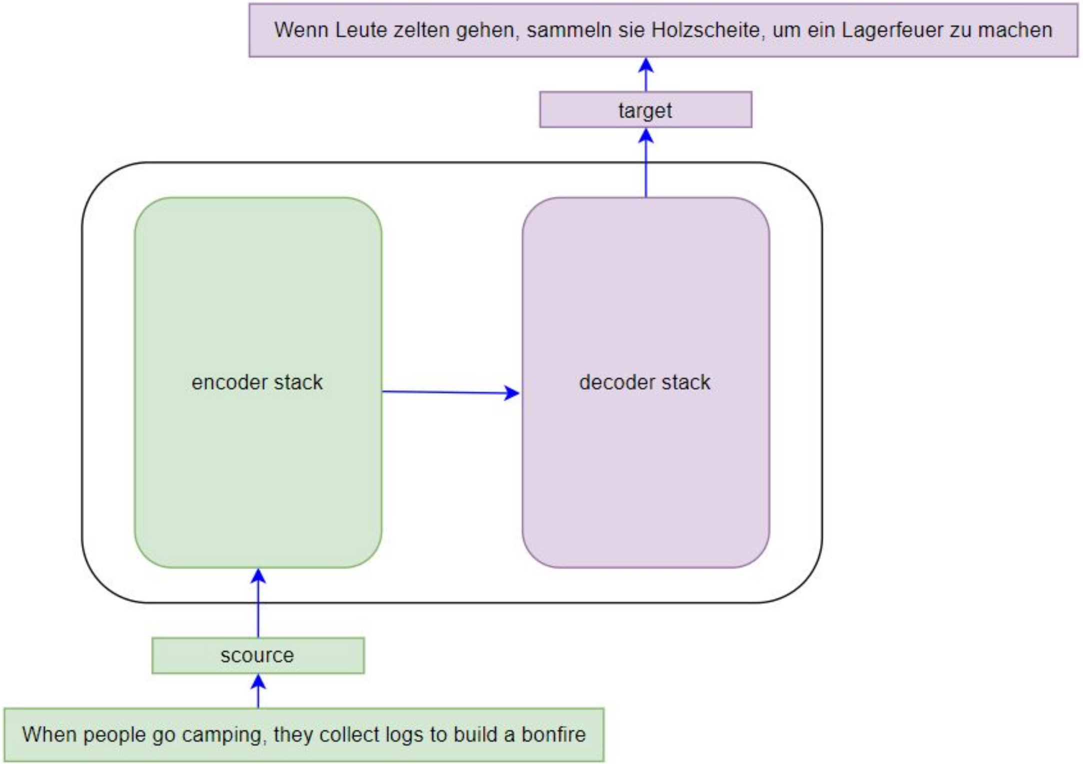
0	0	-inf
0	0	0
0	-inf	-inf

Machine Translation



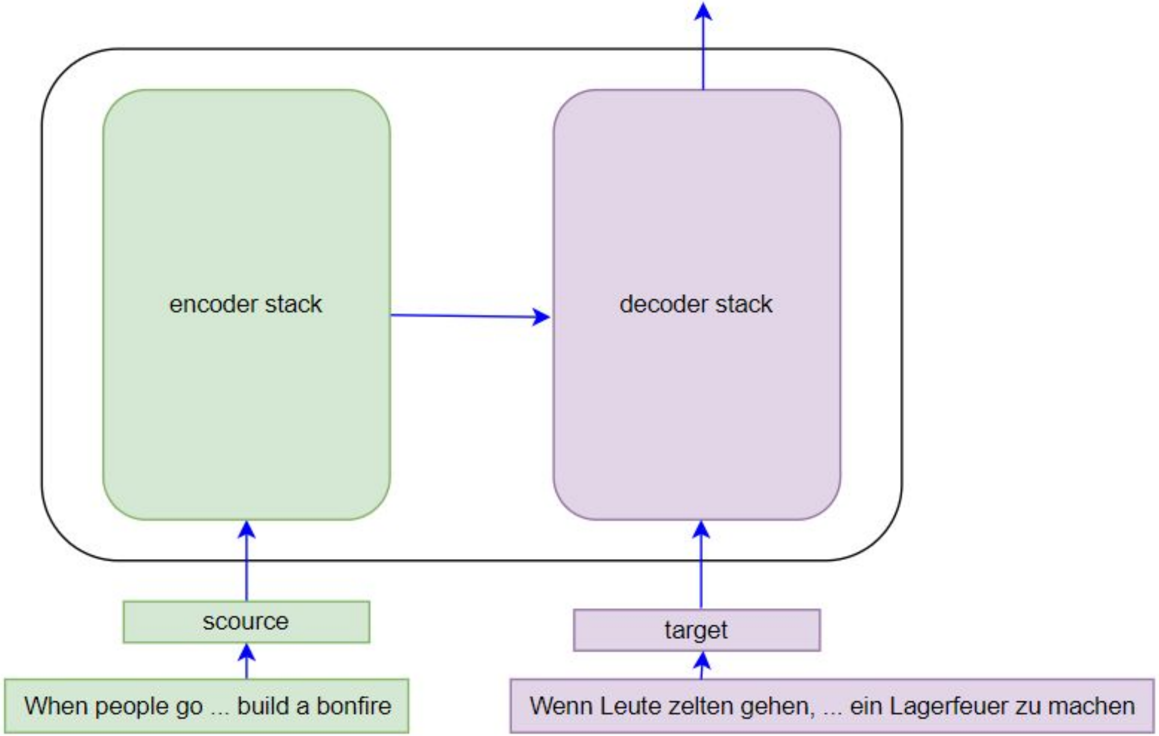
The Transformer Model

At Testing Time:

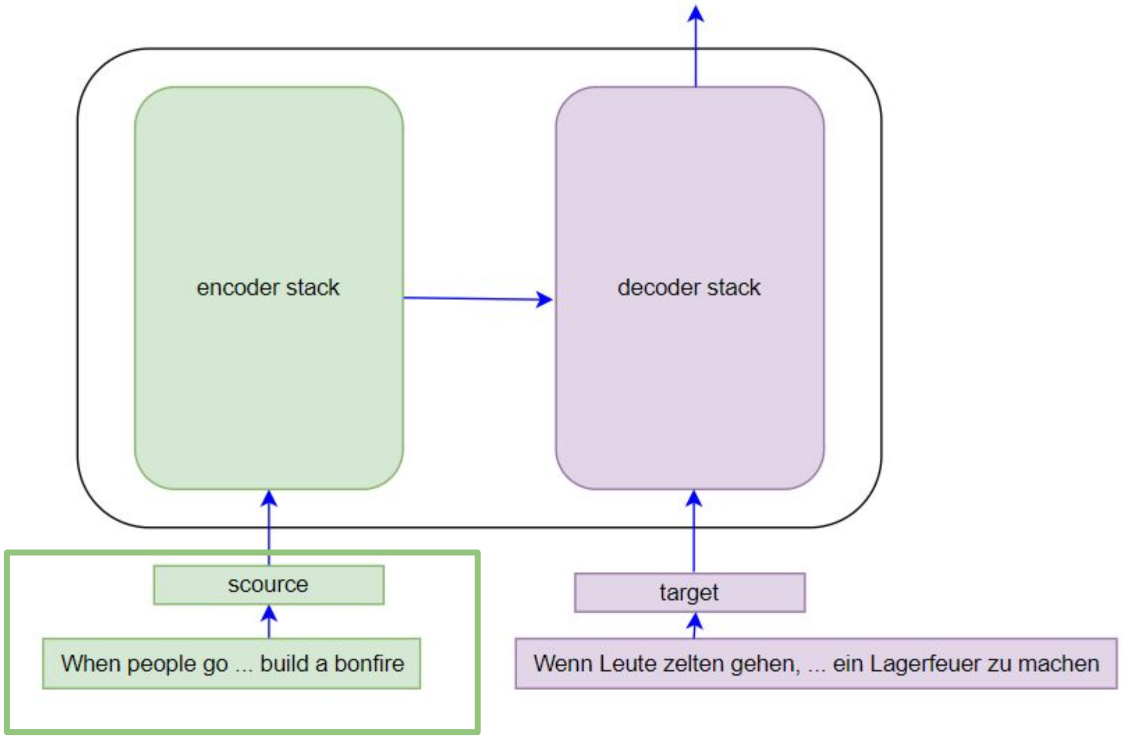


The Transformer Model

At Training Time:

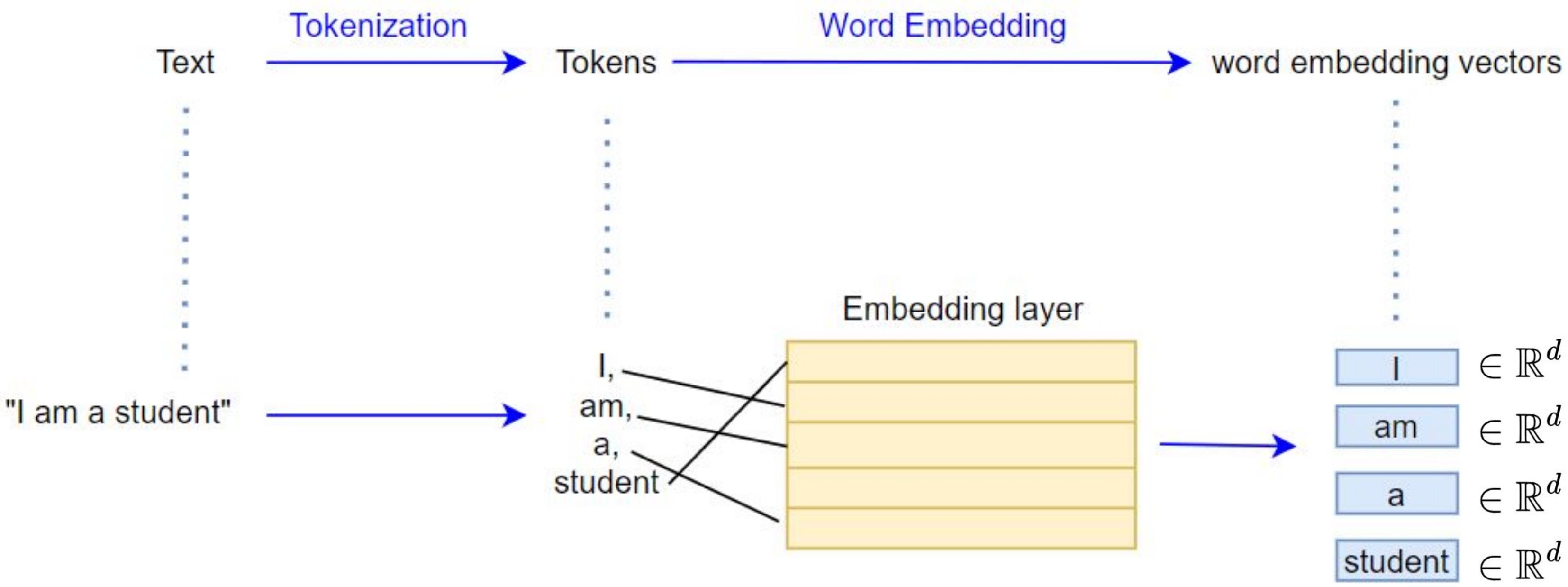


Input Embedding



1. word embedding
2. positional embedding

Word Embedding

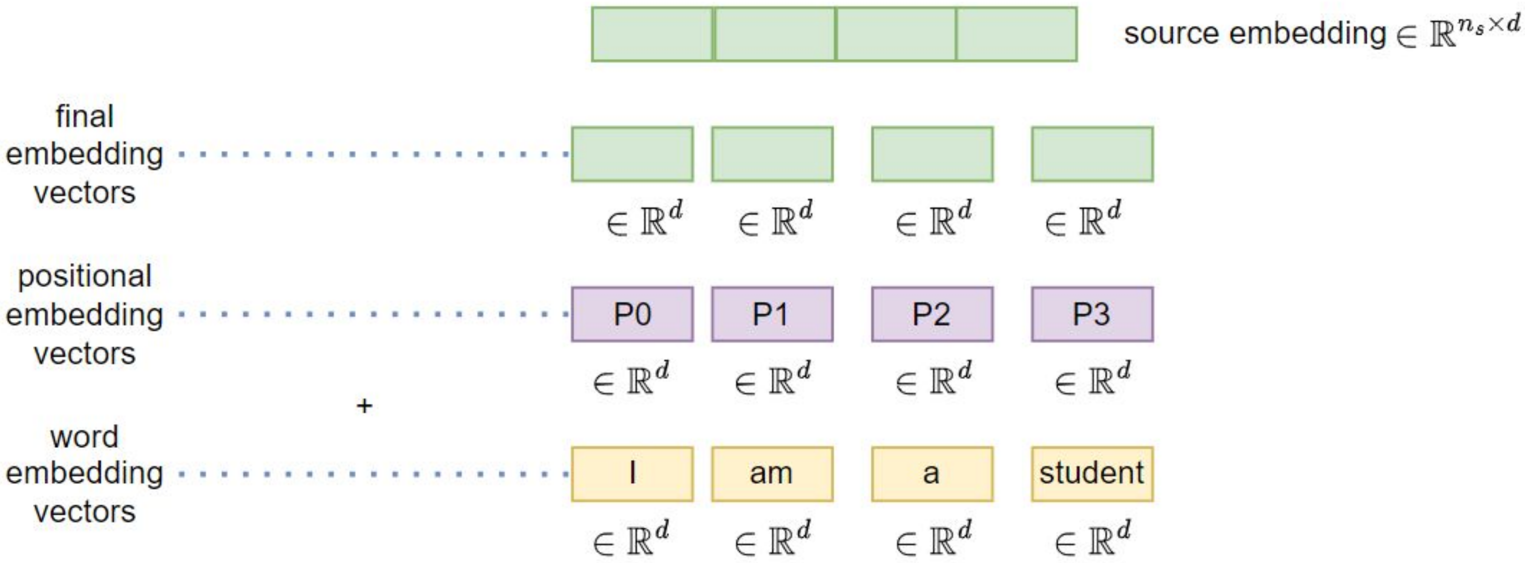


Positional Embedding

order of words matters in some language

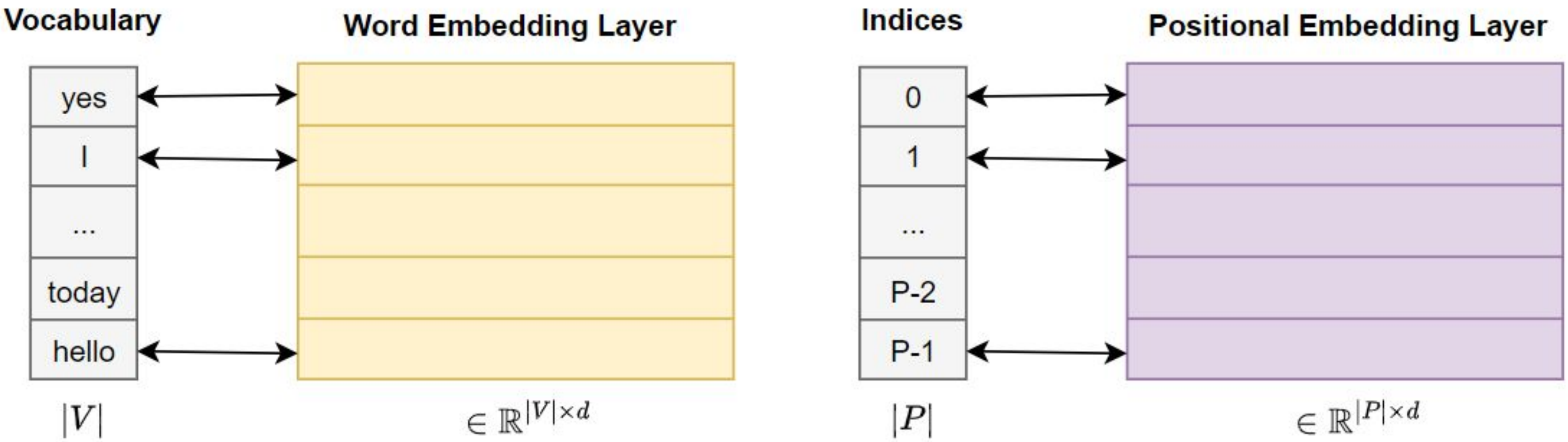
A attacked B.

B attacked A.



Positional Embedding

Way 1: learned



P is a pre-decoded maximum length
The model cannot accept length > P, unless trimmed

Positional Embedding

Way 2: calculated

n_s Total input sequence length

P0 P1 P2 P3 $\in \mathbb{R}^{n_s \times d}$

k The k-th position in total input sequence

P1 $\in \mathbb{R}^d$

j j-th number in d-dimensional vector

P_{10}	P_{11}	P_{12}	...	$P_{1(d-1)}$ $\in \mathbb{R}^d$
0	1	2		d-1
even	odd	even	...	
$2i$	$2i+1$	$2i$		

i Helps denote even and odd positions

Even elements $$P(k, 2i) = sin(\frac{k}{n^{2i/d}})$$

n user defined large scalar, = 10,000 in paper

Odd elements $$P(k, 2i+1) = cos(\frac{k}{n^{2i/d}})$$

Positional Embedding

Even elements $$P(k, 2i) = sin(\frac{k}{n^{2i/d}})$$

n user defined large scalar, = 10,000 in paper

Odd elements $$P(k, 2i+1) = cos(\frac{k}{n^{2i/d}})$$

P0 $\in \mathbb{R}^4$

$k = 0$

$n = 100$

P_{00}	$j = 0, i = 0$	$P(k, 0) = sin(0) = 0$
P_{01}	$j = 1, i = 0$	$P(k, 1) = cos(0) = 1$
P_{02}	$j = 2, i = 1$	$P(k, 2) = sin(0) = 0$
P_{03}	$j = 3, i = 1$	$P(k, 3) = cos(0) = 1$

P0 = [0,1,0,1]

P1 $\in \mathbb{R}^4$

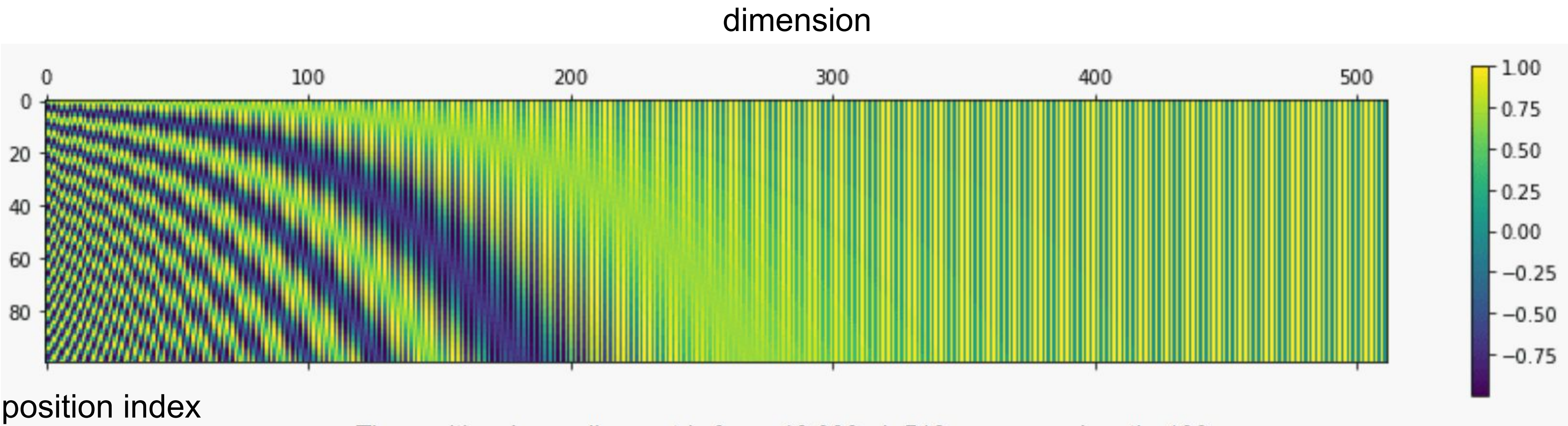
$k = 1$

$n = 100$

P_{10}	$j = 0, i = 0$	$P(k, 0) = sin(1/n^{\frac{0}{4}}) = sin(1) \approx 0.84$
P_{11}	$j = 1, i = 0$	$P(k, 1) = cos(1/n^{\frac{0}{4}}) = cos(1) \approx 0.54$
P_{12}	$j = 2, i = 1$	$P(k, 2) = sin(1/100^{\frac{2}{4}}) = sin(1/10) \approx 0.10$
P_{13}	$j = 3, i = 1$	$P(k, 3) = cos(1/100^{\frac{2}{4}}) = cos(1/10) \approx 1.0$

P1 = [0.84,0.54,0.1,1]

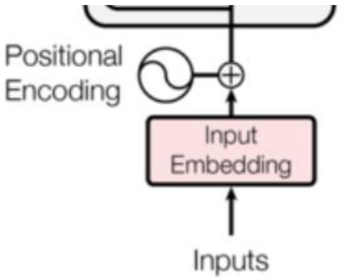
Positional Embedding



The positional encoding matrix for n=10,000, d=512, sequence length=100

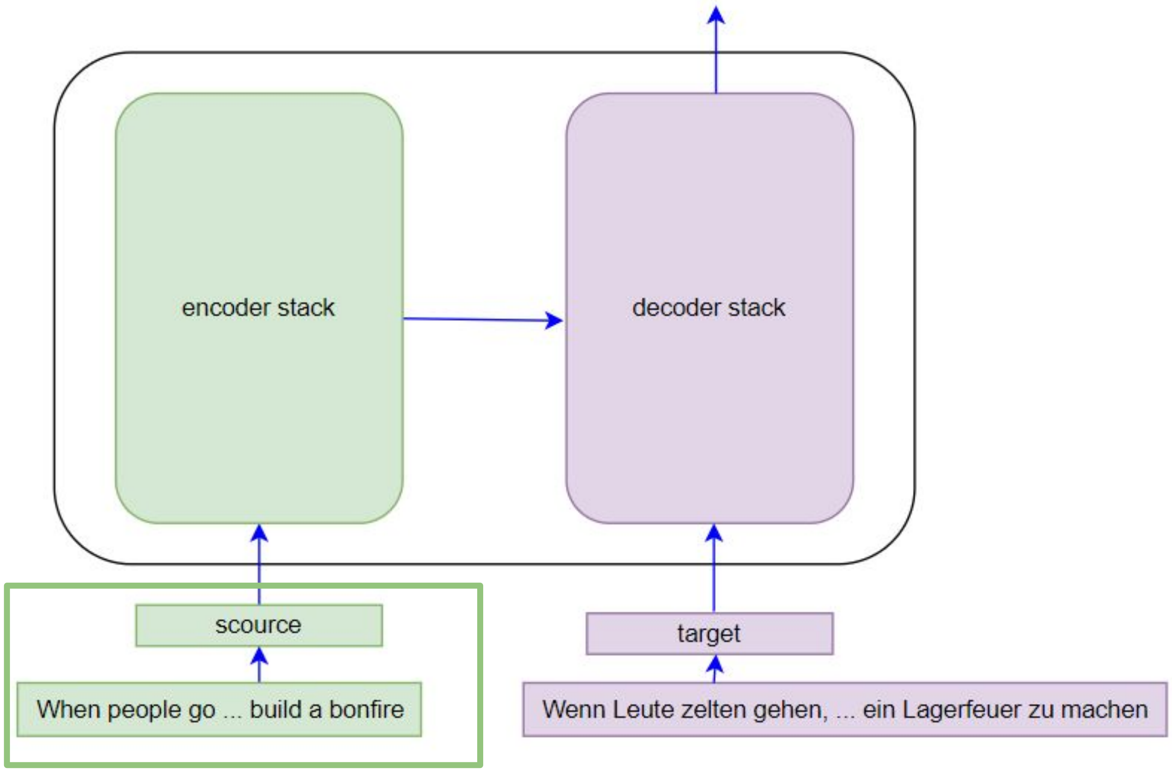
image source

Input Embedding

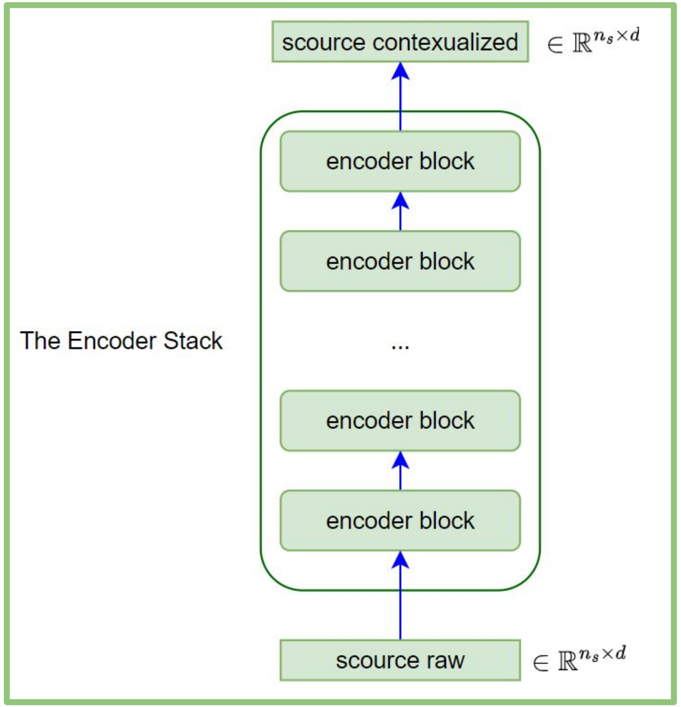


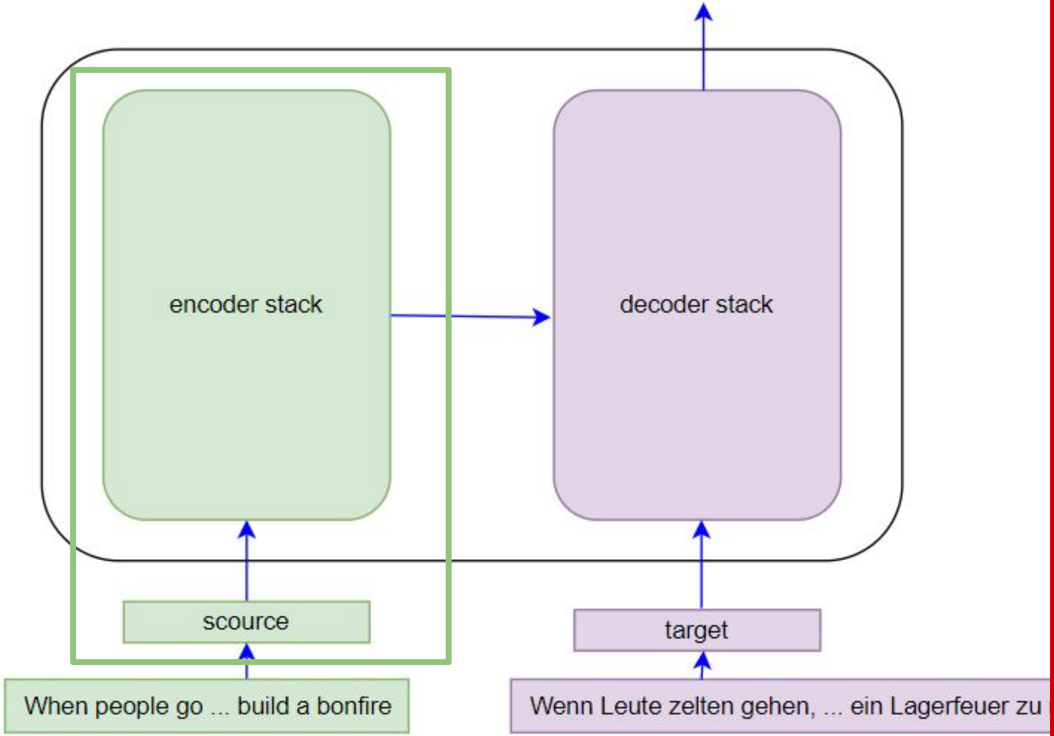
In original paper

1. word embedding
2. positional embedding

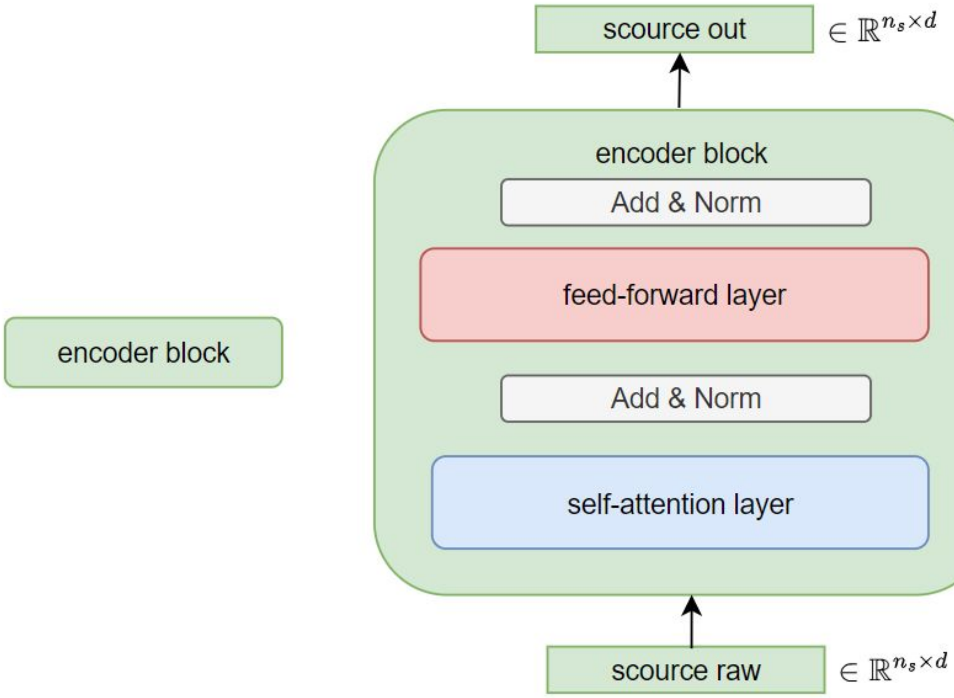


The Encoder Stack

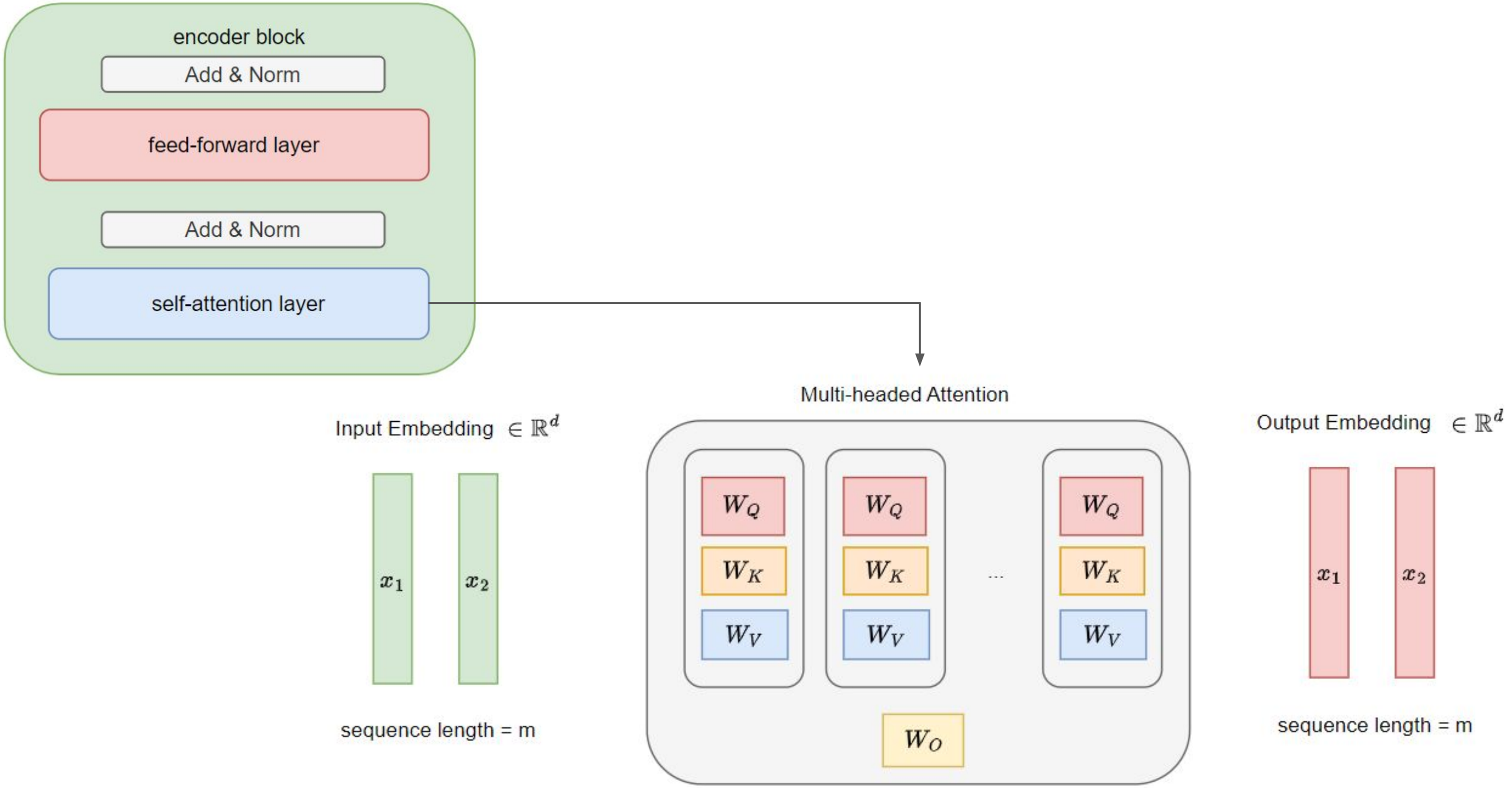




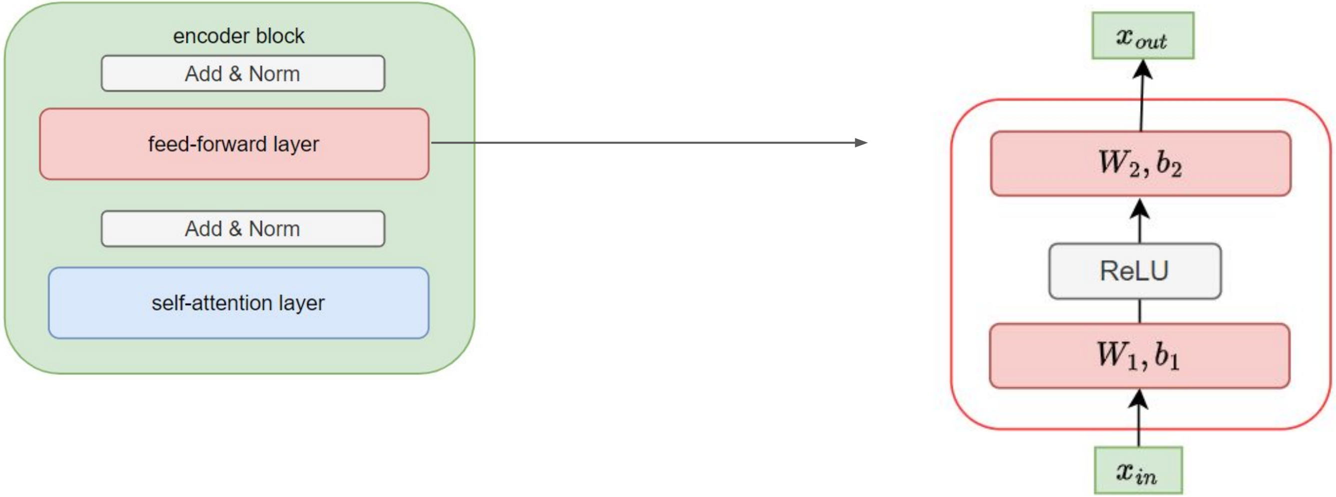
The Encoder Block



Multi-headed Attention

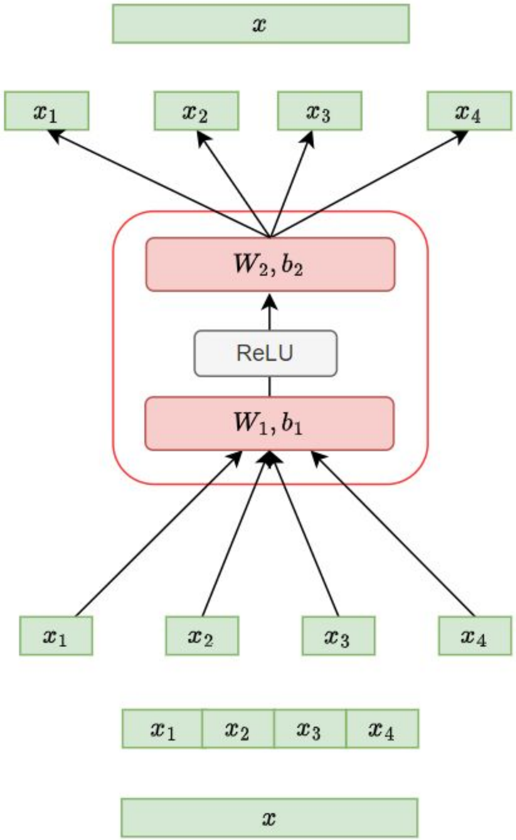


Position-wise Feed-forward Layer



$$x_{out} = (ReLU(x_{in}^T W_1 + b_1))W_2 + b_2$$

Position-wise Feed-forward Layer

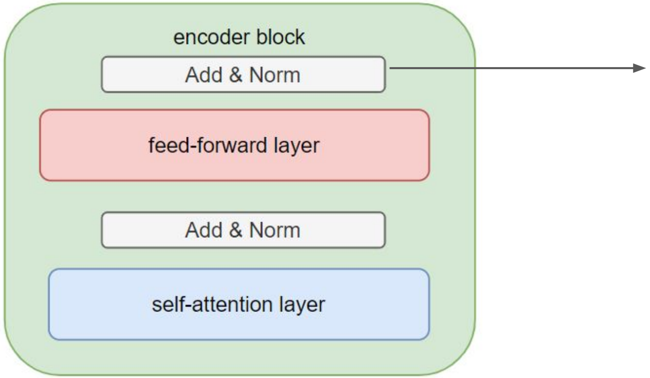


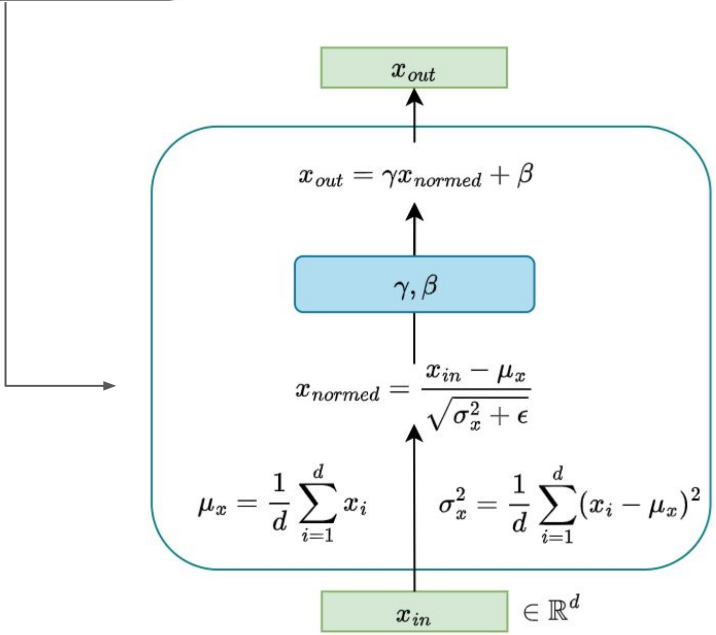
$$x_{out} = (ReLU(x_{in}^T W_1 + b_1))W_2 + b_2$$

“x_in” is each individual word

not a whole (n_s) * d sequence

Add & Norm



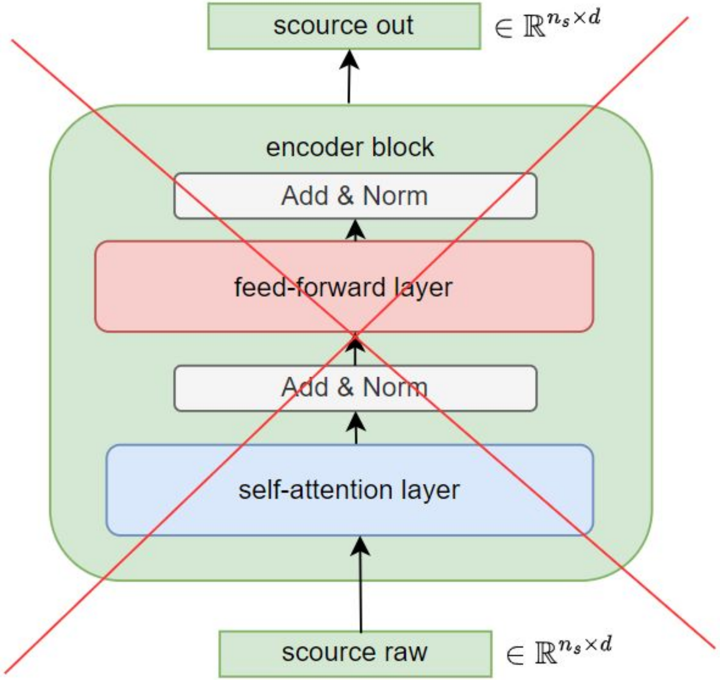


ϵ : A small number to prevent denominator becomes too small, and underflow due to limited precision

γ, β : learnable parameters

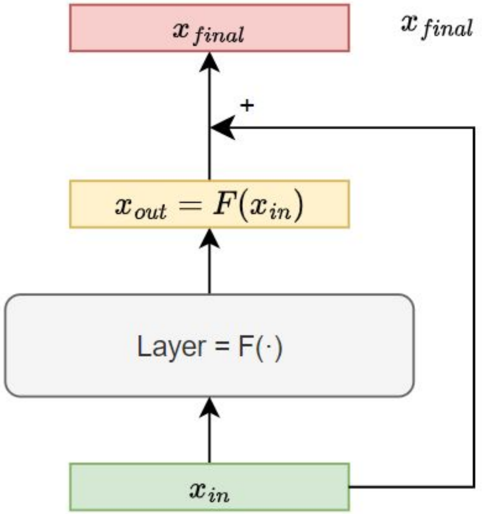
Ba et. al. 2016. Layer Normalization

The Encoder Block



NOT simply pass through one layer after another

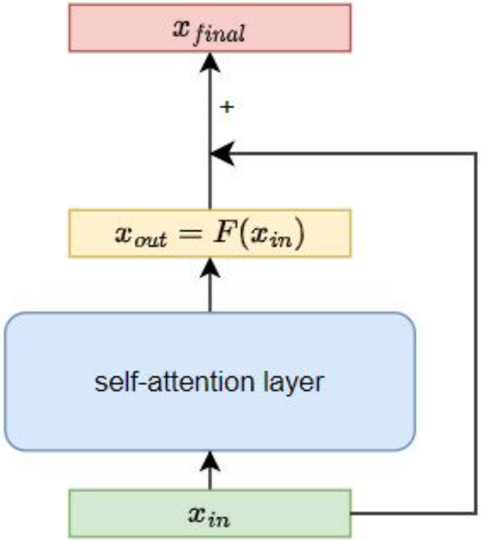
Residual Connection

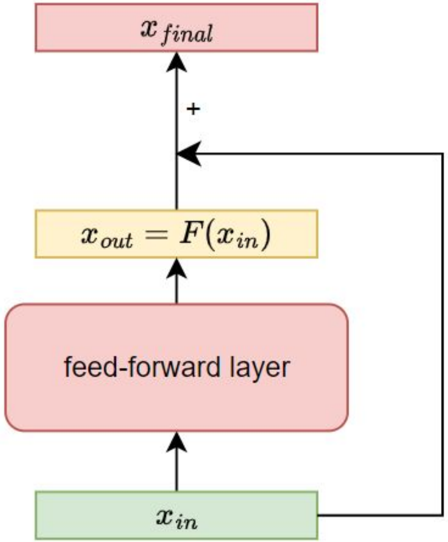


$x_{final} = x_{out} + x_{in}$

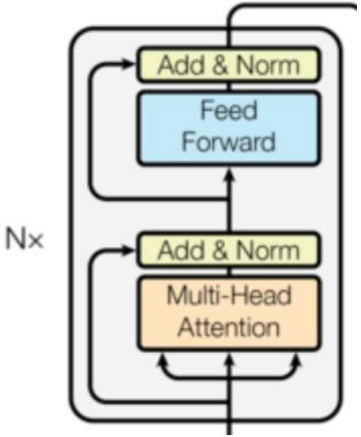
He et. al. 2015. Deep Residual Learning for Image Recognition

Residual Connection

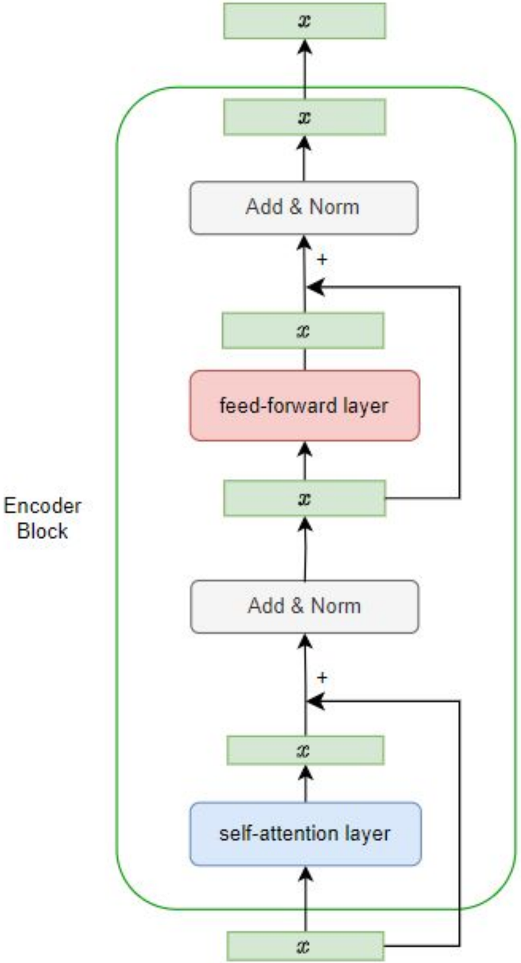




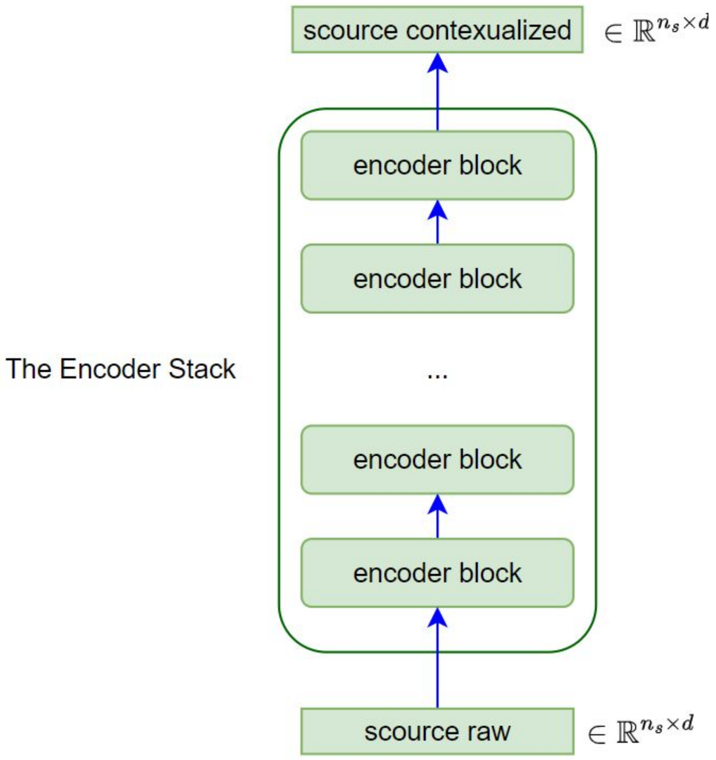
Full encoder block

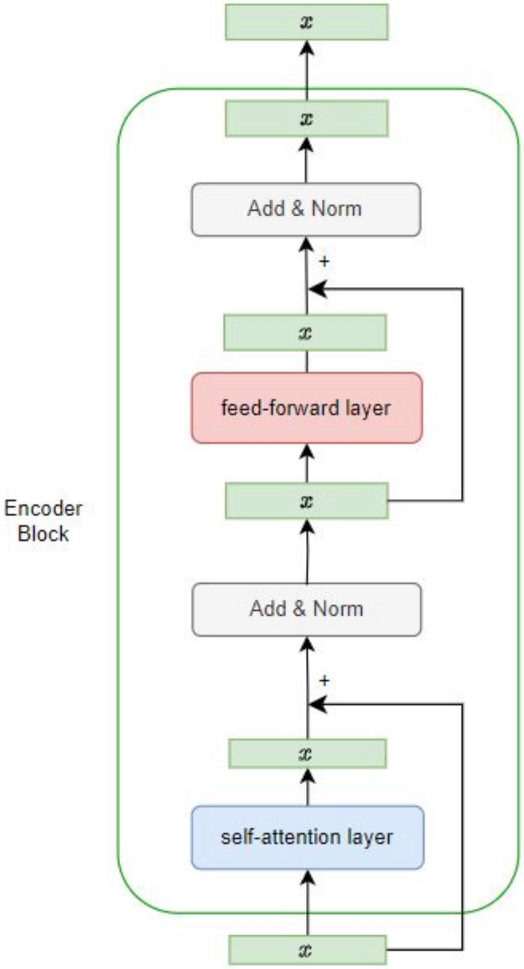


In original paper

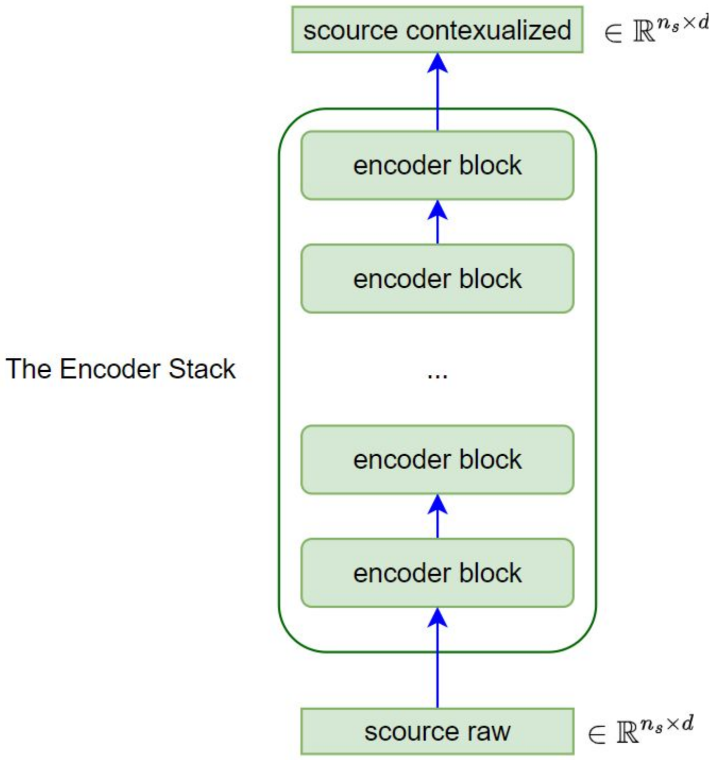


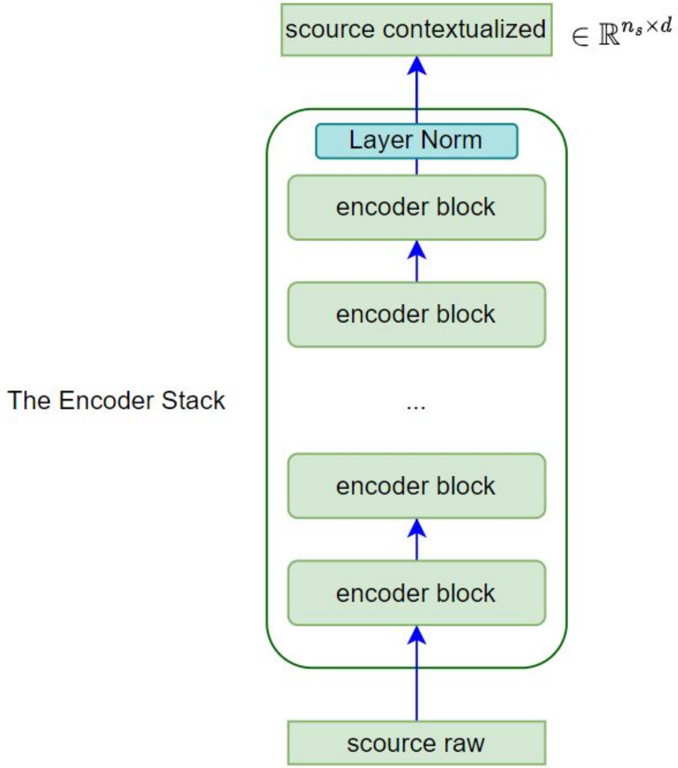
Full encoder stack





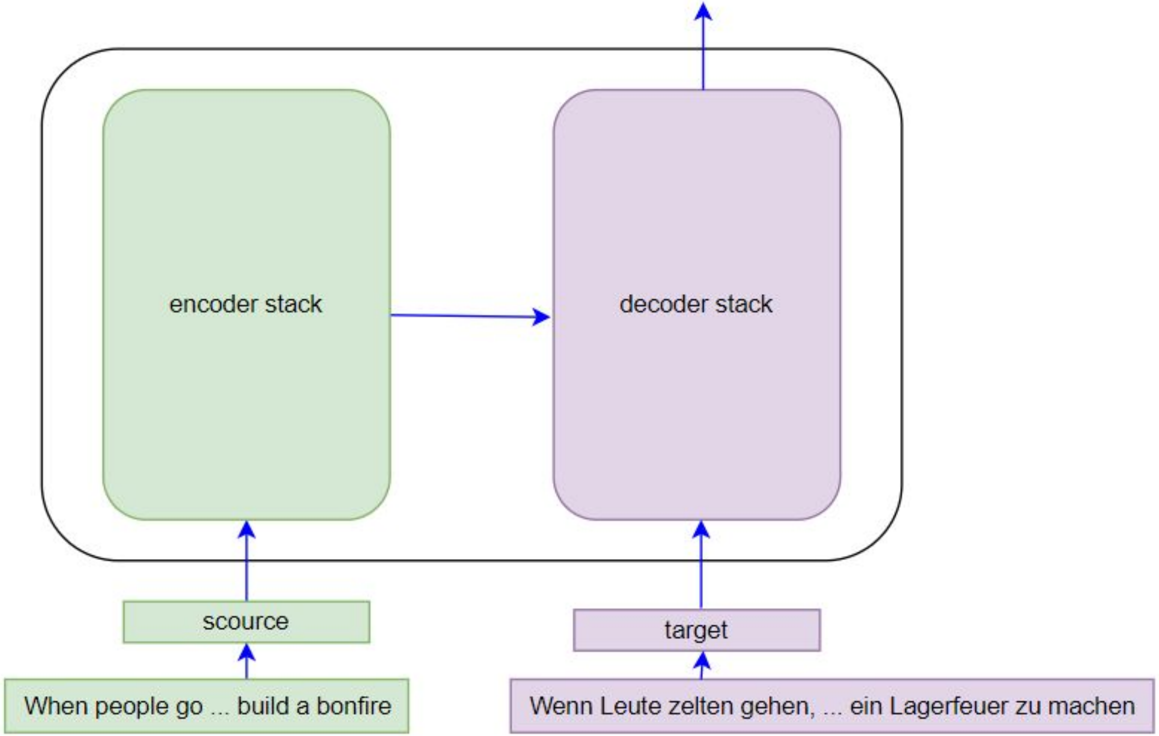
Full encoder stack





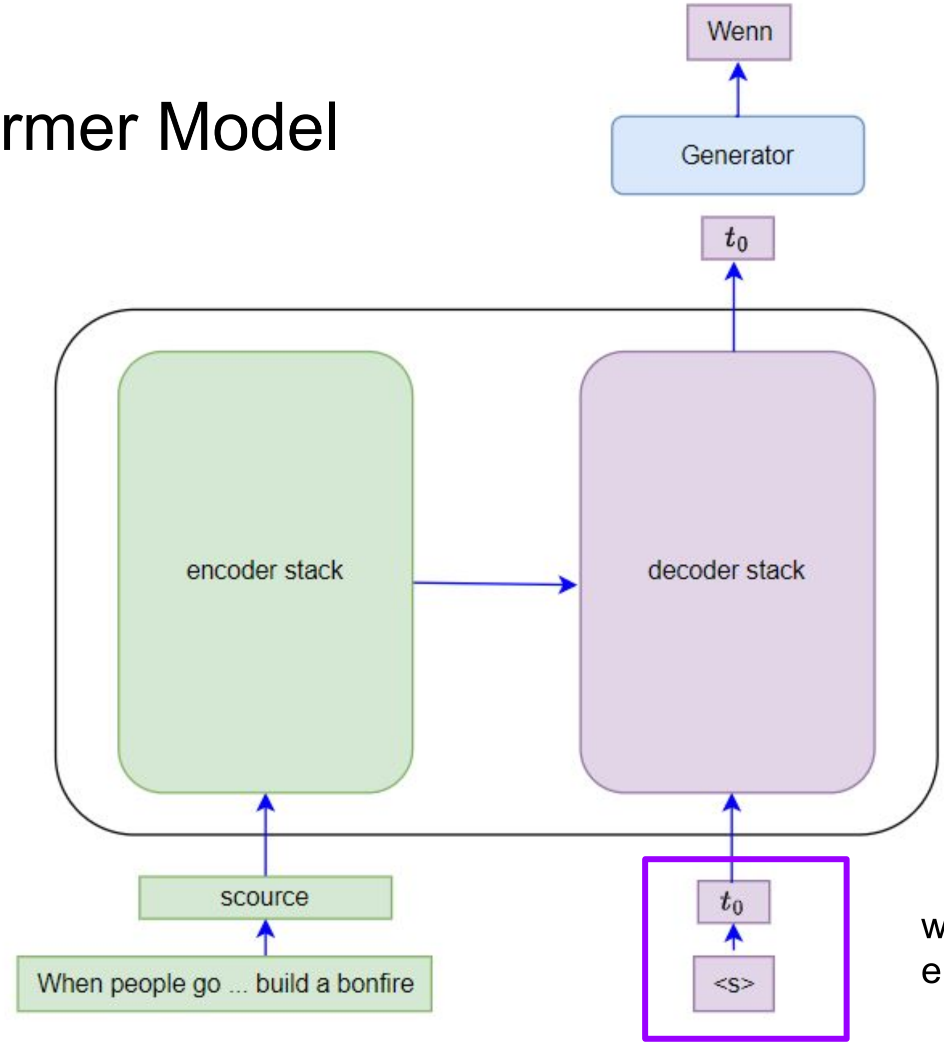
The Transformer Model

At Training Time:



The Transformer Model

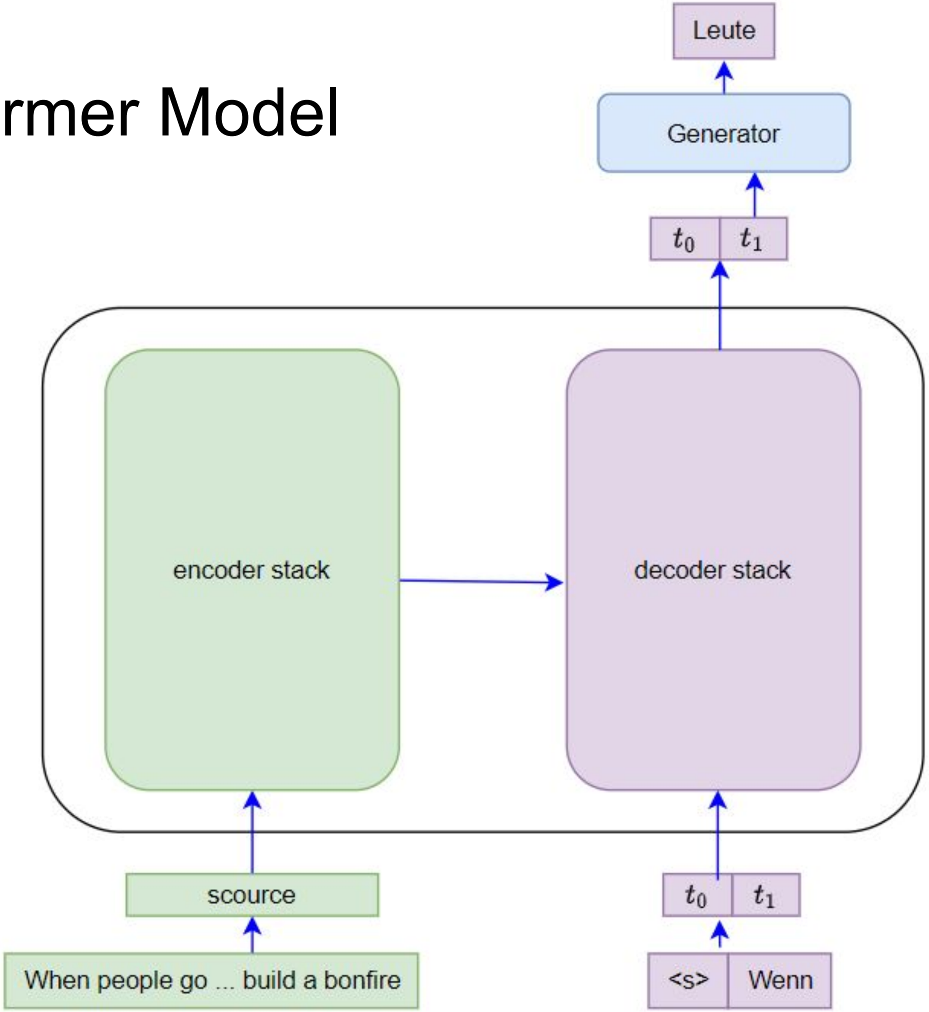
At Testing Time:



word + pos embedding

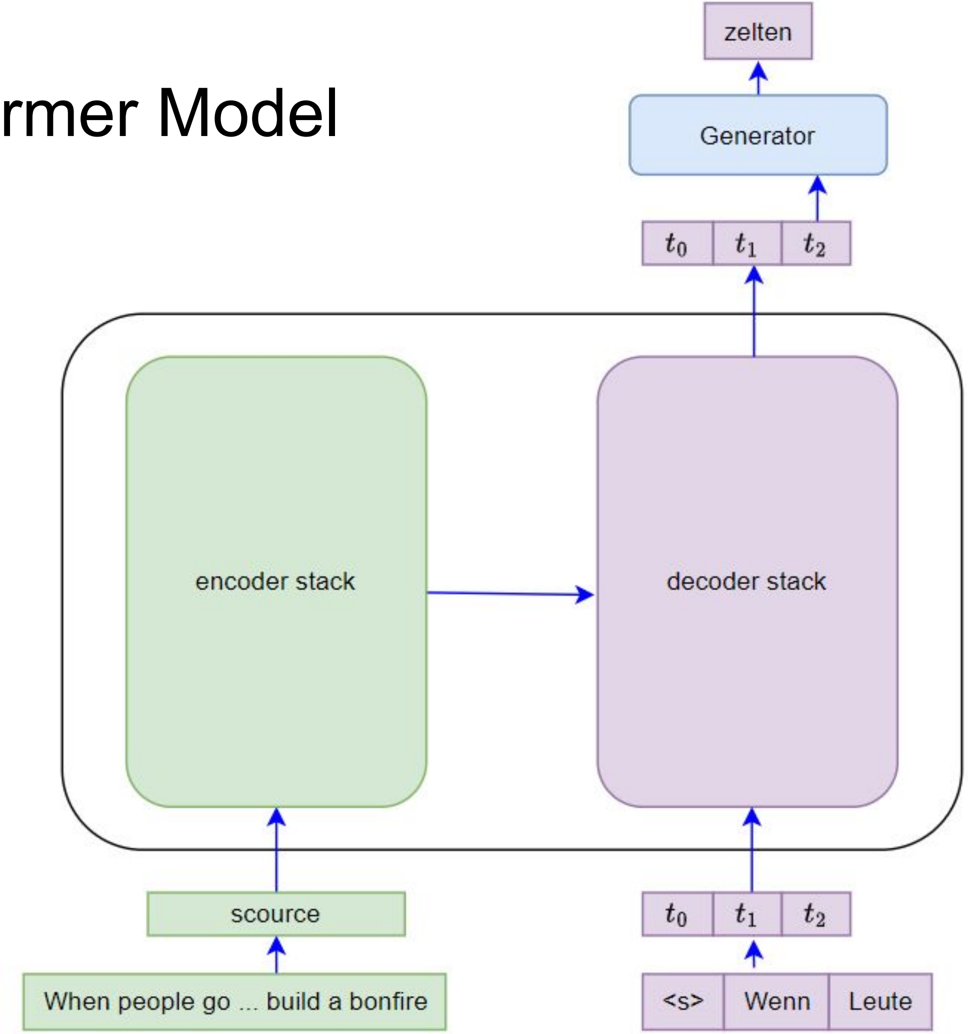
The Transformer Model

At Testing Time:



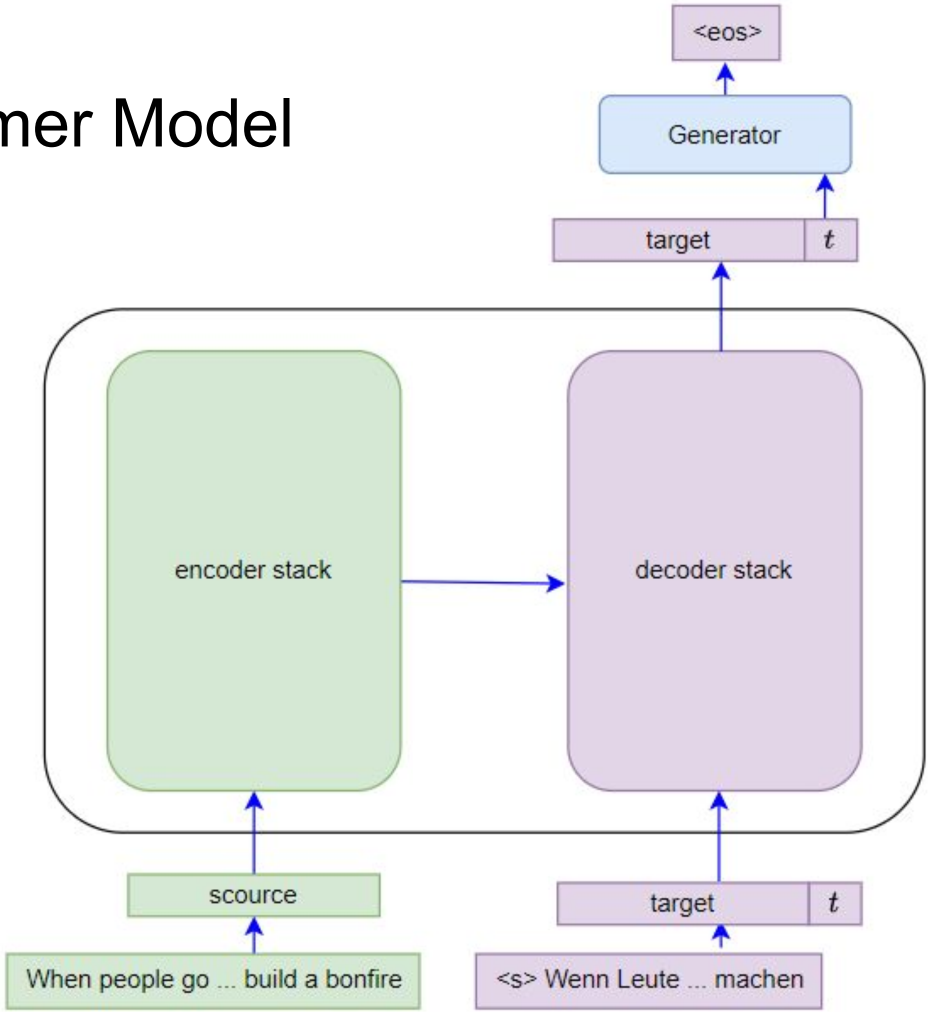
The Transformer Model

At Testing Time:



The Transformer Model

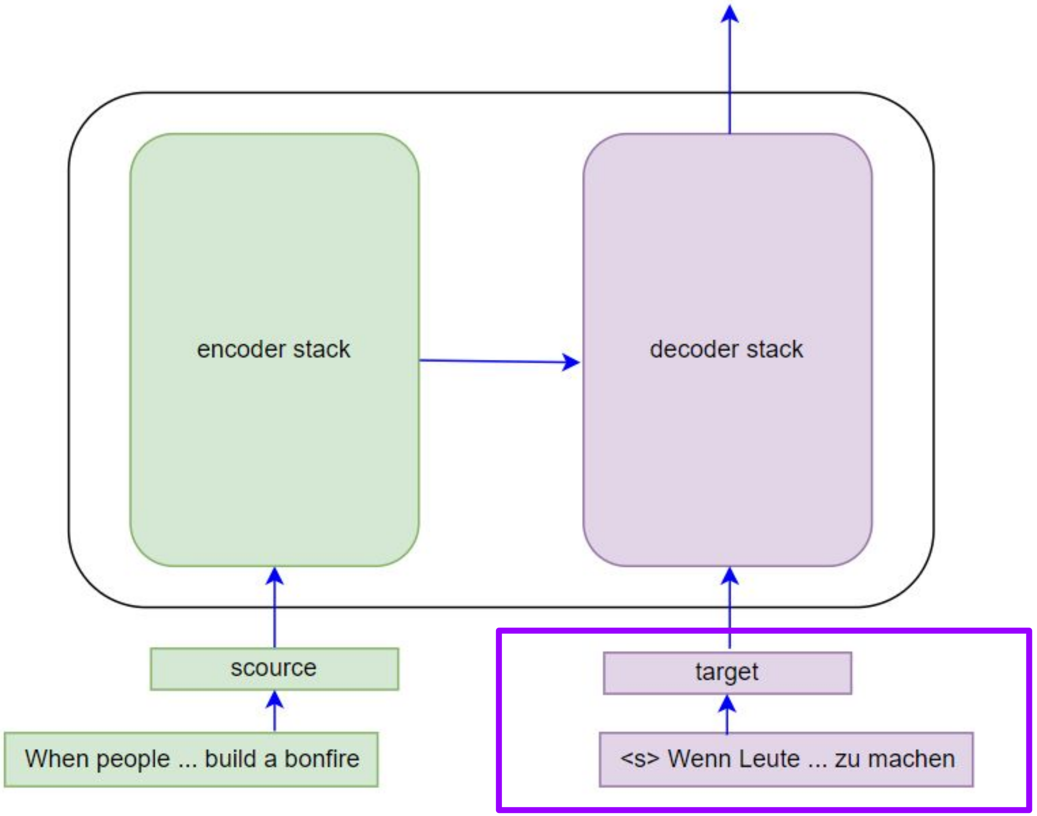
At Testing Time:



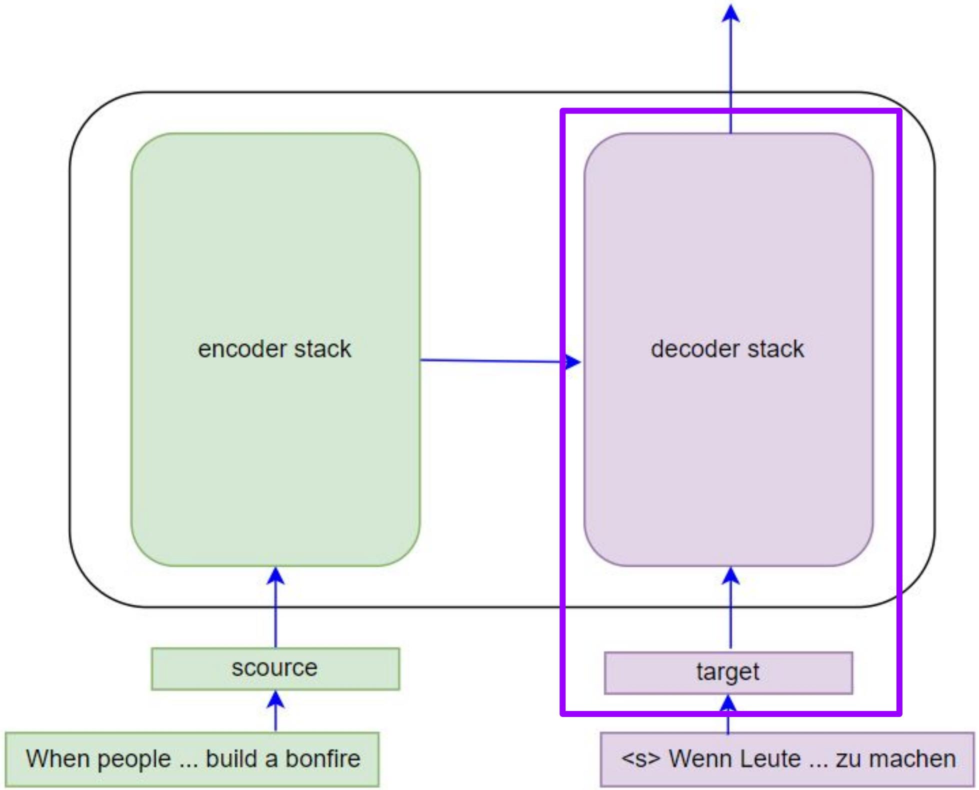
Target Input Embedding

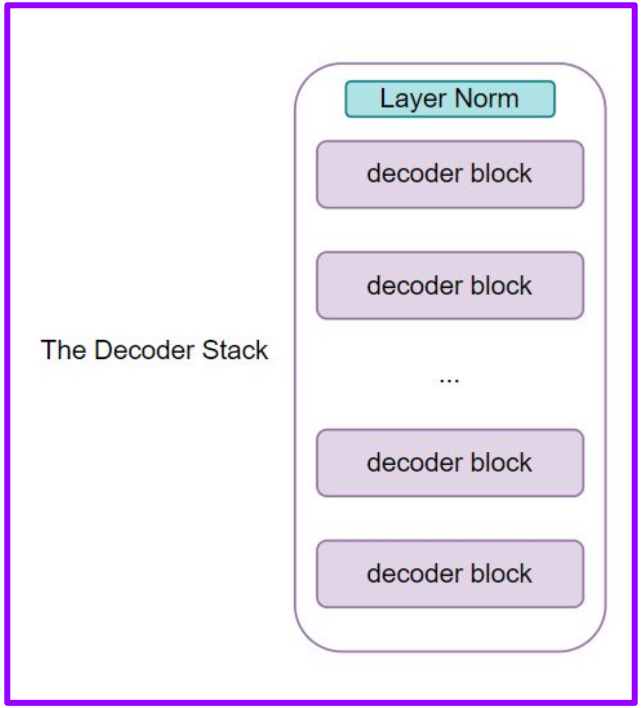
Add one special character at beginning
To shift every word one position behind

1. word embedding
2. positional embedding



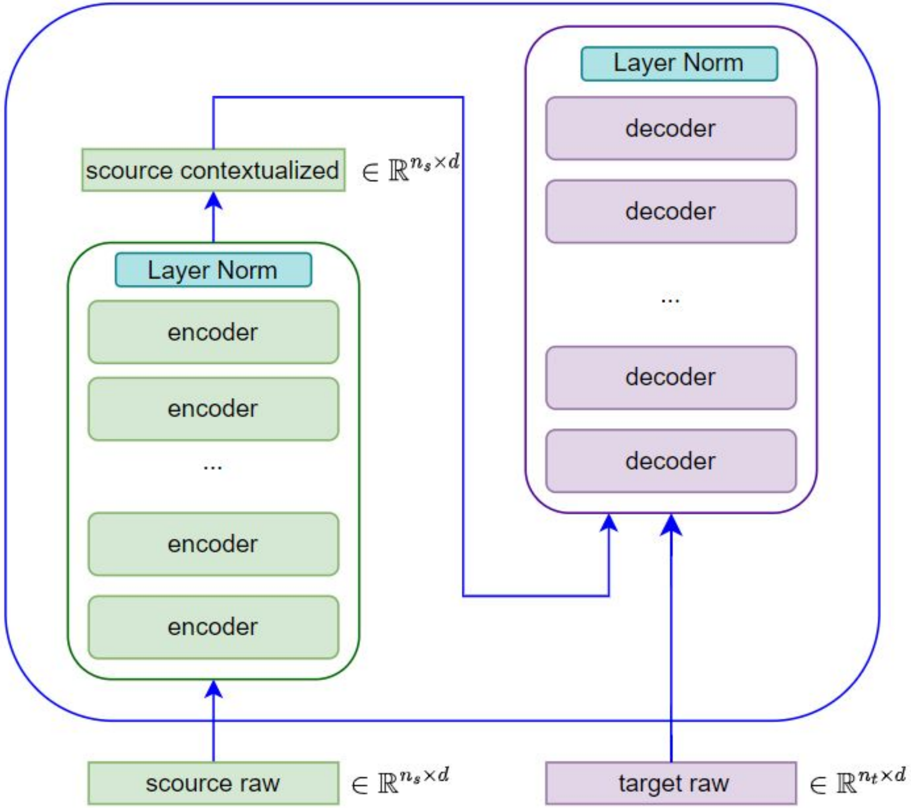
The Decoder Stack



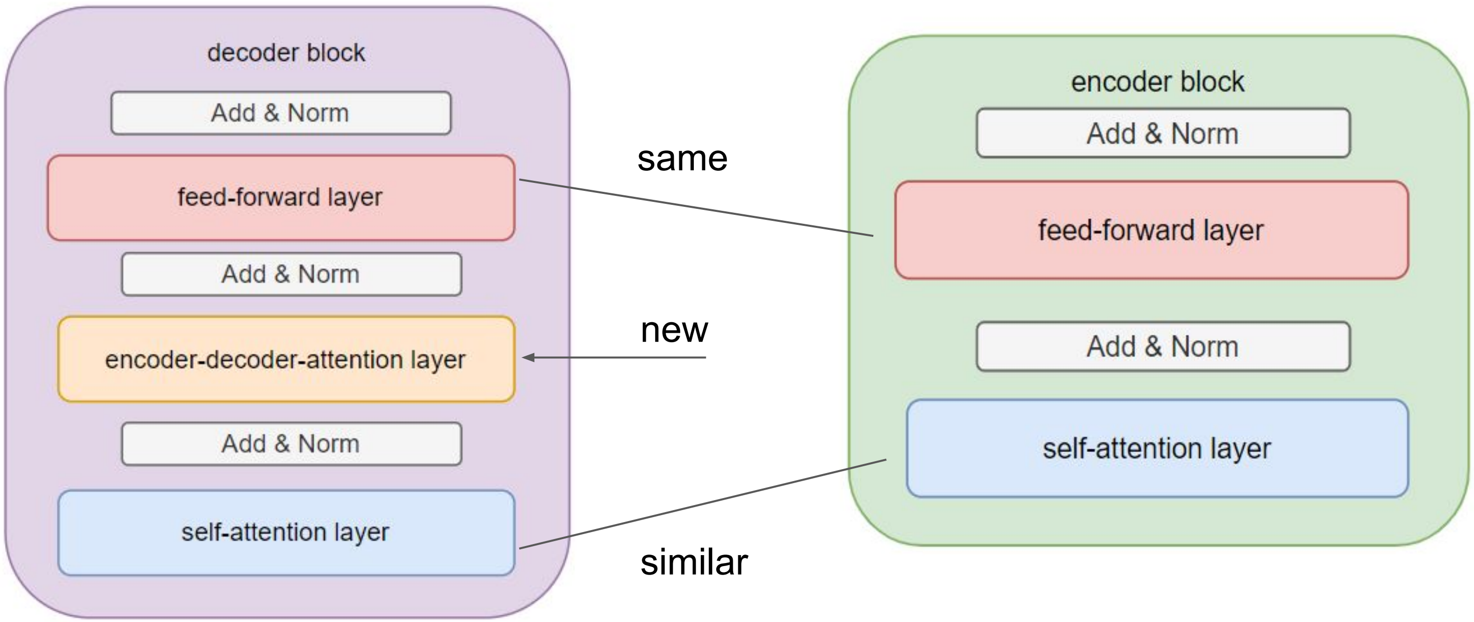


The Decoder Stack

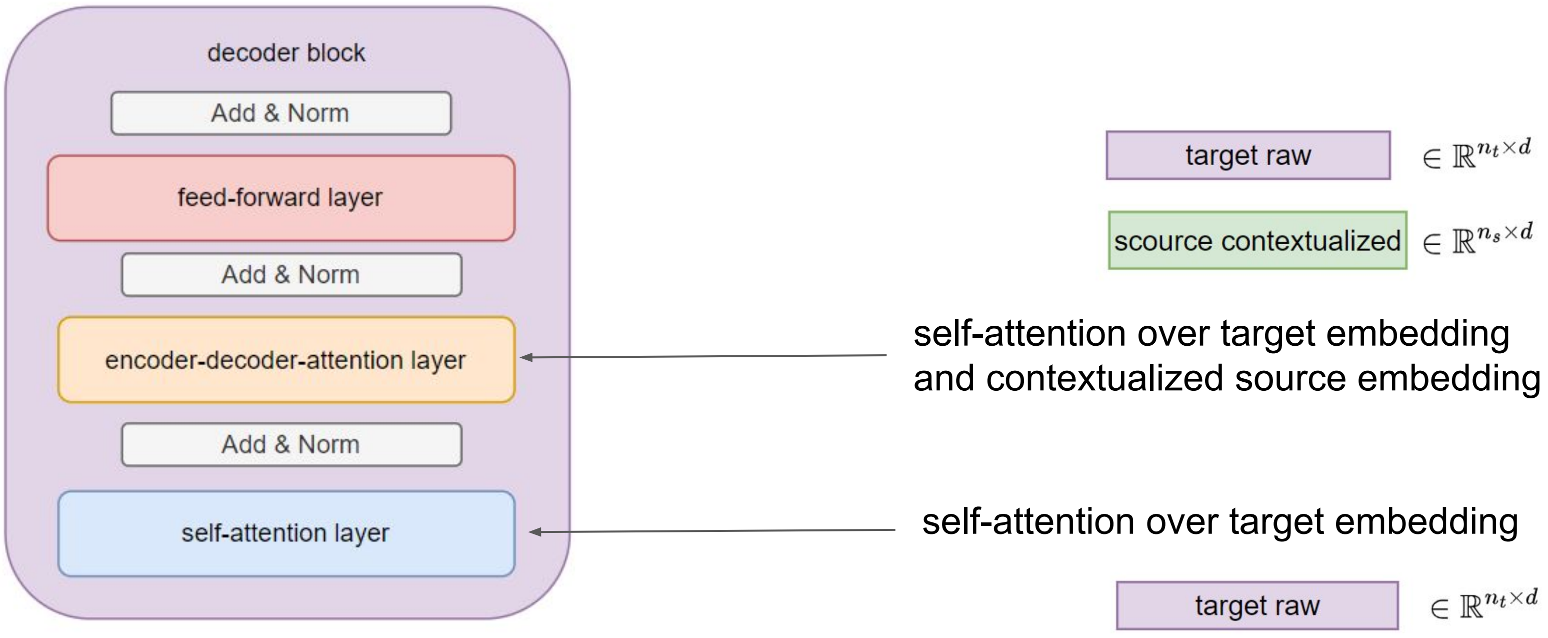
Training Time:



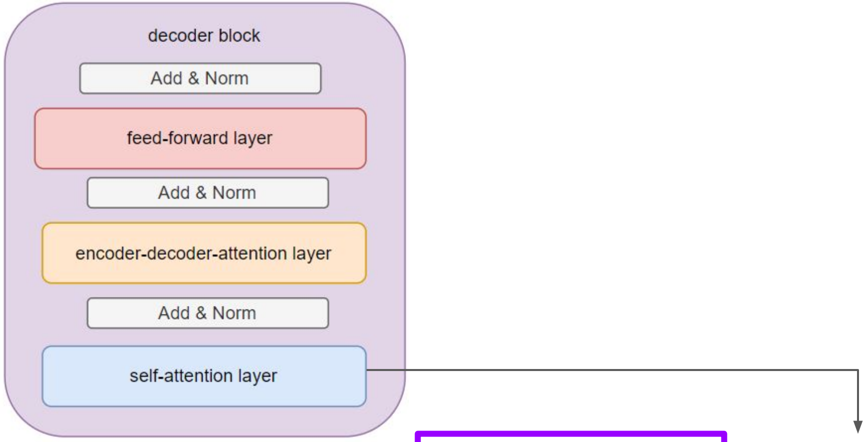
The Decoder Block



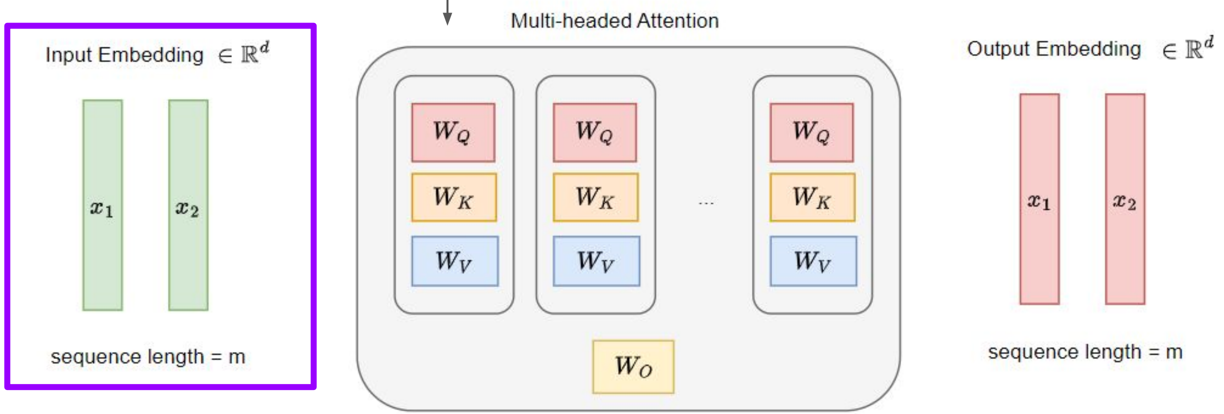
The Decoder Block



The Decoder Block



But uses a special attention mask



target embedded

The Decoder Block Self-attention Mask

Good	morning	[PAD]

Recall: use attention mask to avoid attention to [PAD] token

0	0	-inf
0	0	-inf
0	0	-inf

Not only [PAD] token, also can be used to mask any position in attention matrix we don’t want

The Decoder Block Self-attention Mask

Every word in decoder self-attention will only attend to words before it and itself

Ich	weiß	nicht

The reason is, **during test time**,
when we have no ground-truth target embedding given,
we will predict each word **one-by-one**, not together

<s> → Ich
<s> Ich → weiß
<s> Ich weiß → nicht

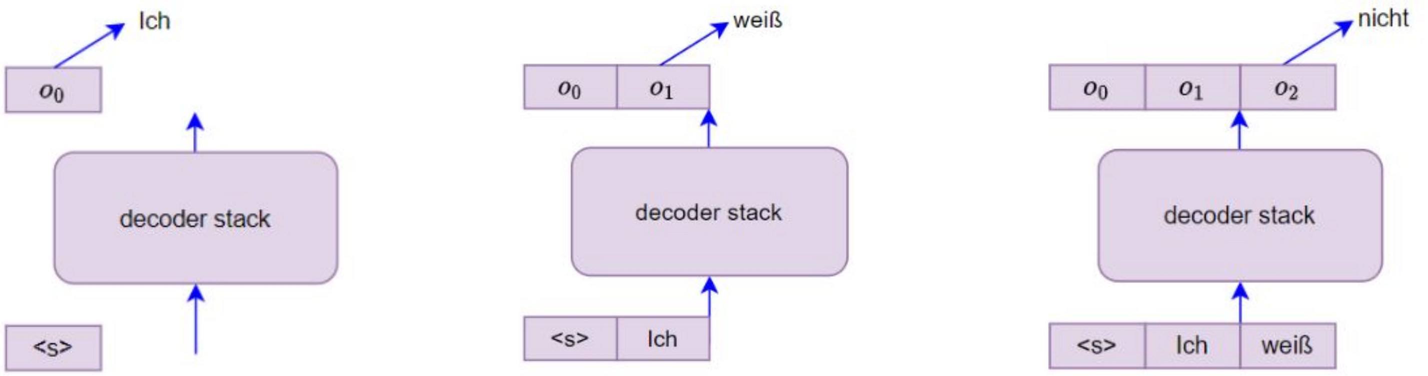
Each word can only see previous words in test time
So we mimic the same thing at training time

$x_1 \rightarrow x_1$	$x_1 \rightarrow x_2$	$x_1 \rightarrow x_3$
$x_2 \rightarrow x_1$	$x_2 \rightarrow x_2$	$x_2 \rightarrow x_3$
$x_3 \rightarrow x_1$	$x_3 \rightarrow x_2$	$x_3 \rightarrow x_3$

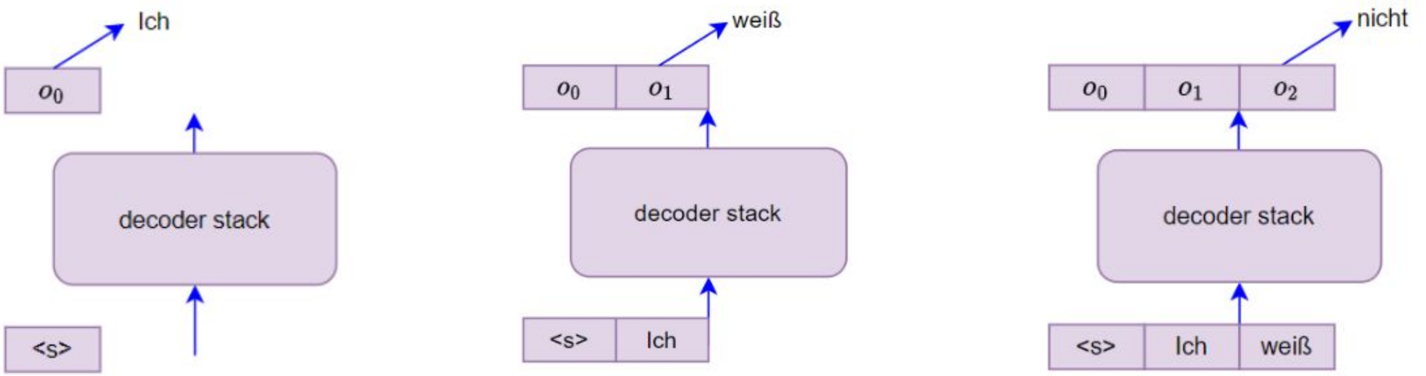
green means attends,
gray means masked

The Decoder Block Self-attention Mask

Testing



Training

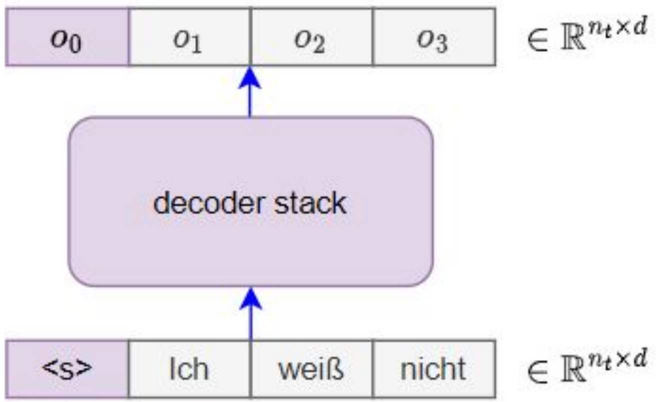


possible,
but want less
training time

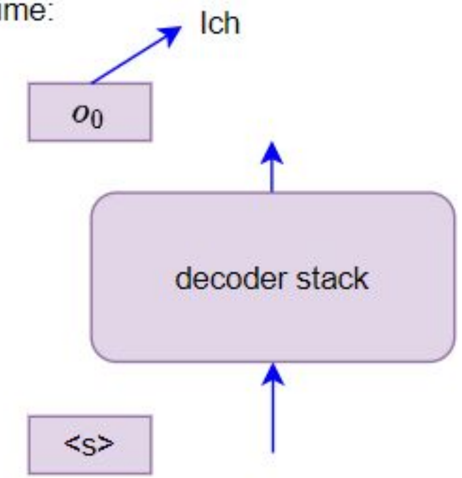
The Decoder Block Self-attention Mask

$x_1 \to x_1$	$x_1 \to x_2$	$x_1 \to x_3$
$x_2 \to x_1$	$x_2 \to x_2$	$x_2 \to x_3$
$x_3 \to x_1$	$x_3 \to x_2$	$x_3 \to x_3$

Training Time:



Testing Time:

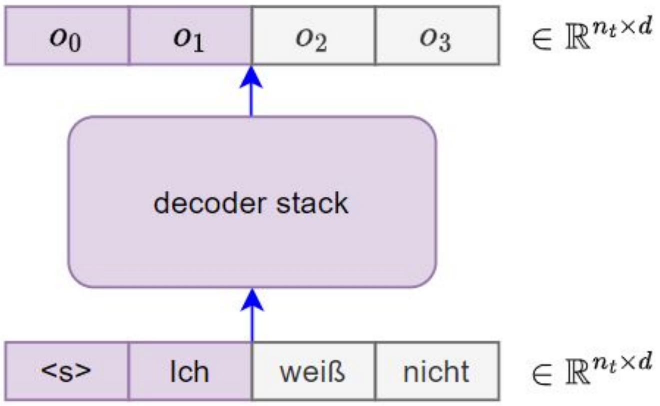


o_0 only attends to start of sentence character

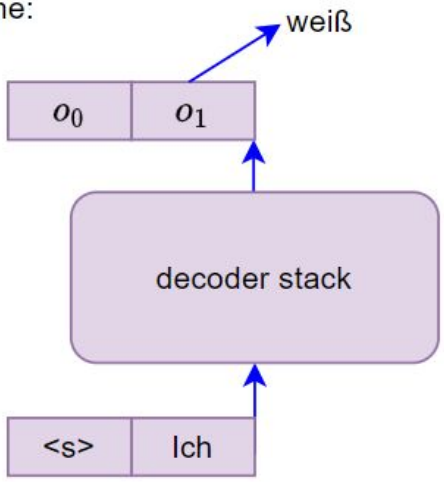
The Decoder Block Self-attention Mask

$x_1 \to x_1$	$x_1 \to x_2$	$x_1 \to x_3$
$x_2 \to x_1$	$x_2 \to x_2$	$x_2 \to x_3$
$x_3 \to x_1$	$x_3 \to x_2$	$x_3 \to x_3$

Training Time:



Testing Time:

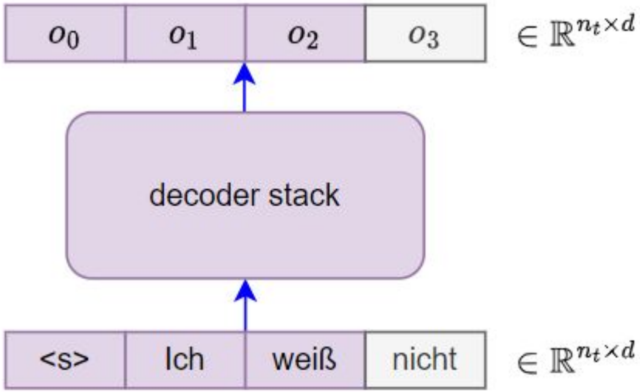


o_1 attends to precious words

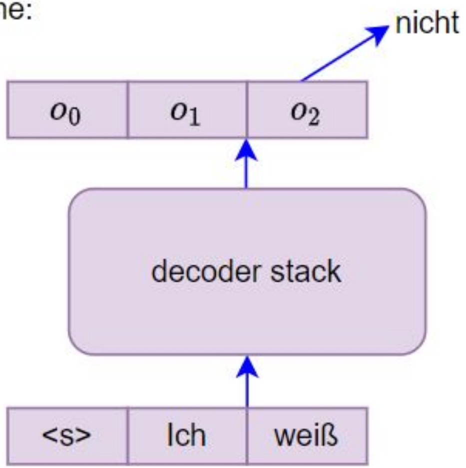
The Decoder Block Self-attention Mask

$x_1 \rightarrow x_1$	$x_1 \rightarrow x_2$	$x_1 \rightarrow x_3$
$x_2 \rightarrow x_1$	$x_2 \rightarrow x_2$	$x_2 \rightarrow x_3$
$x_3 \rightarrow x_1$	$x_3 \rightarrow x_2$	$x_3 \rightarrow x_3$

Training Time:

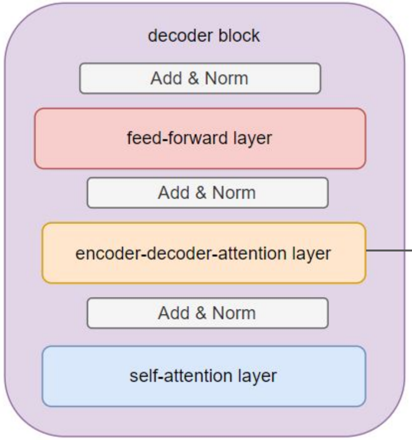


Testing Time:

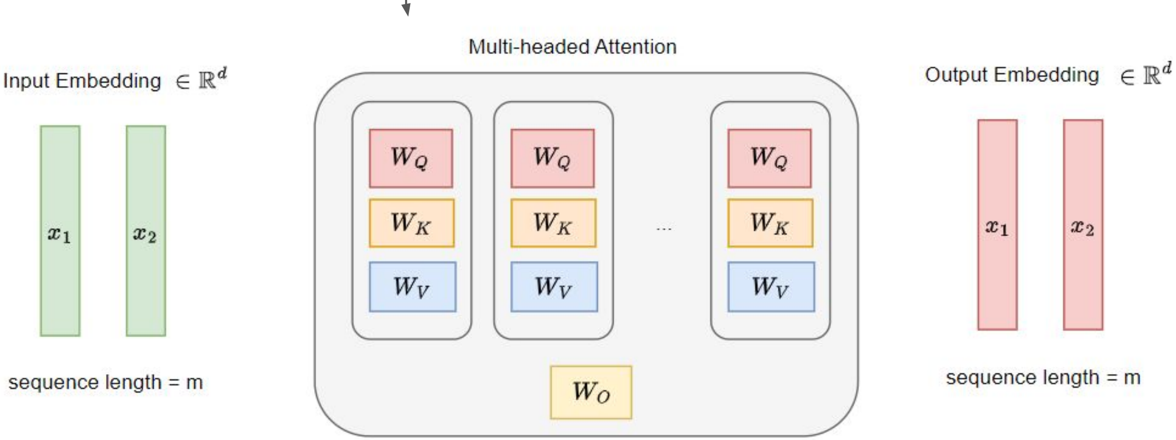


o_2 attends to precious words

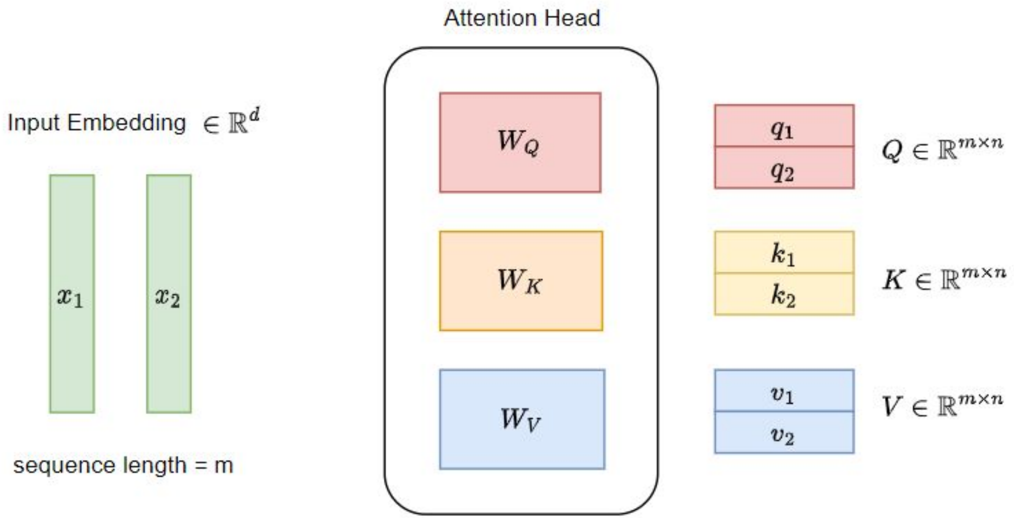
The encoder-decoder attention



But compures QKV differently



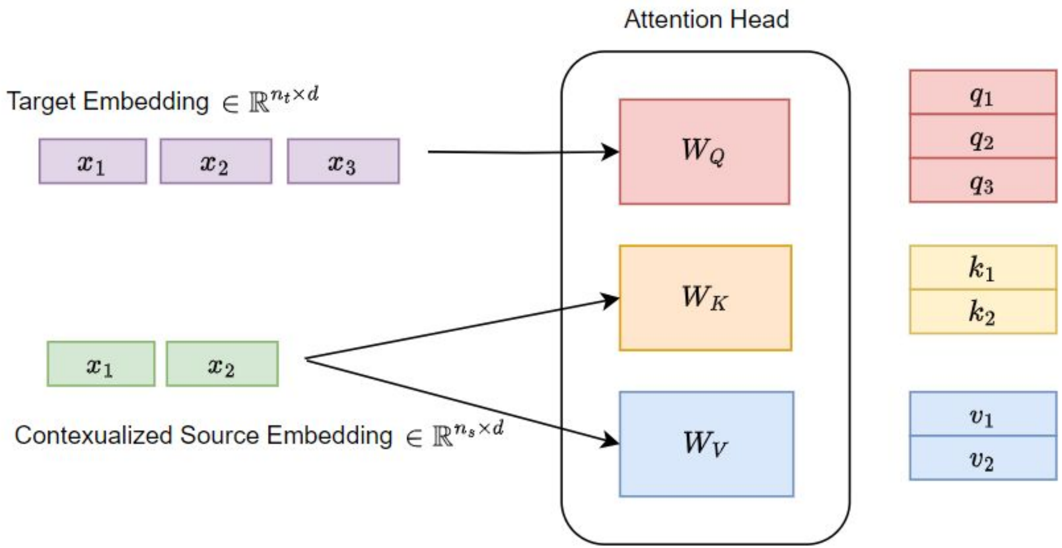
The encoder-decoder attention



In encoder-self-attention, QKV are computed from source embedding

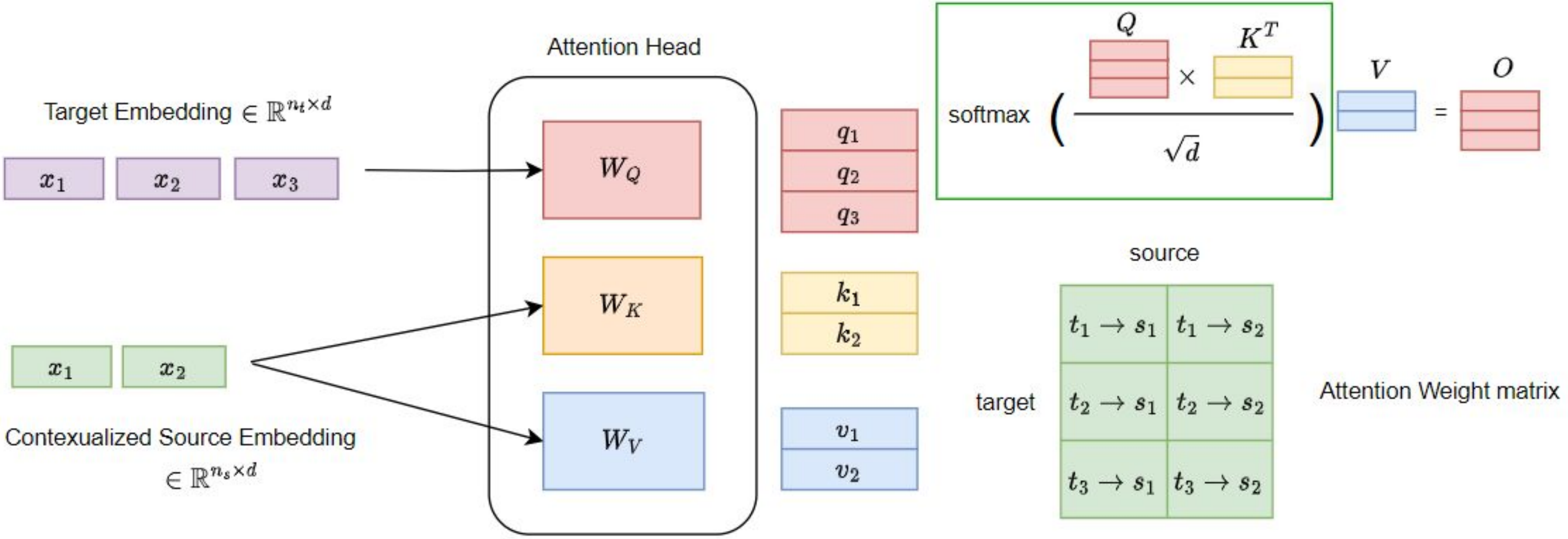
In decoder-self-attention, QKV are computed from target embedding

The encoder-decoder attention



In encoder-decoder-attention,
Q is computed from target embedding
KV are computed from contextualized source embedding

The encoder-decoder attention



The encoder-decoder attention

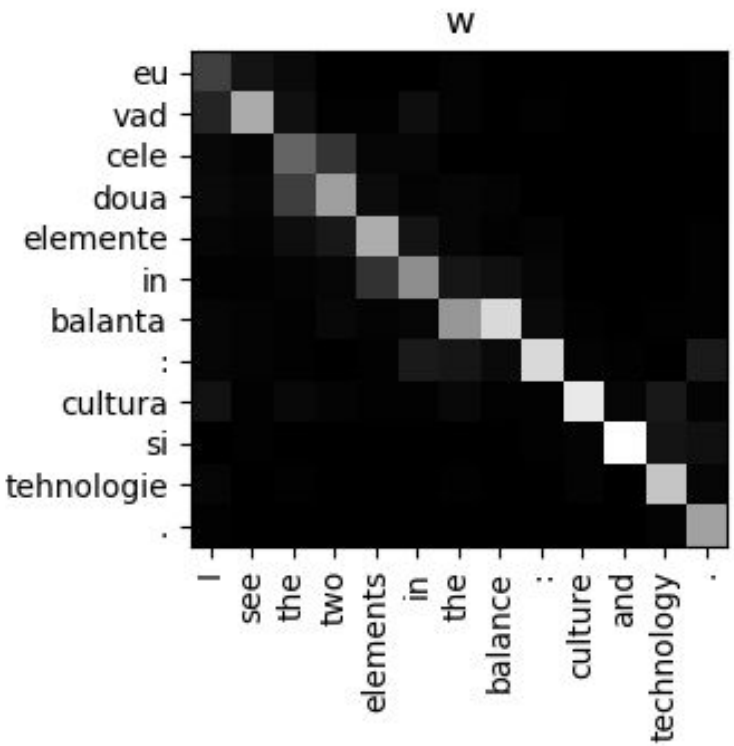
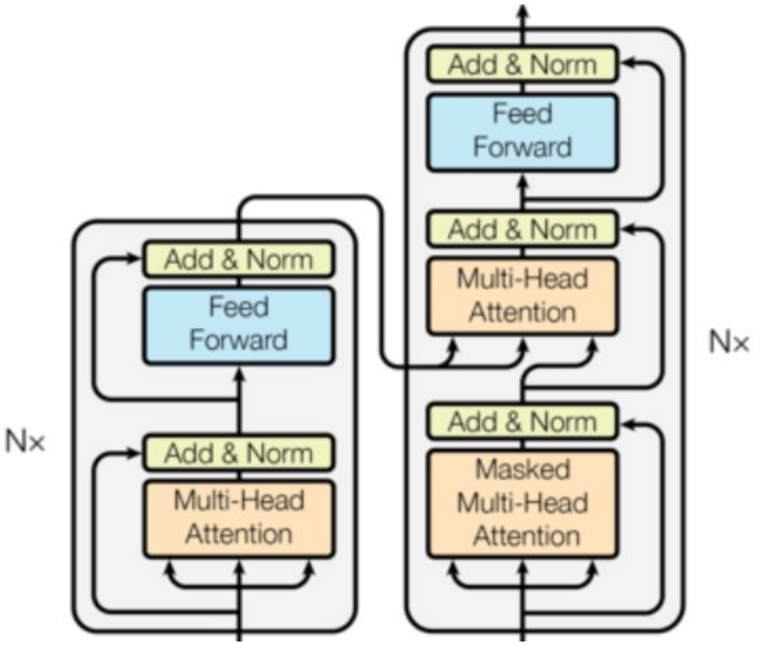
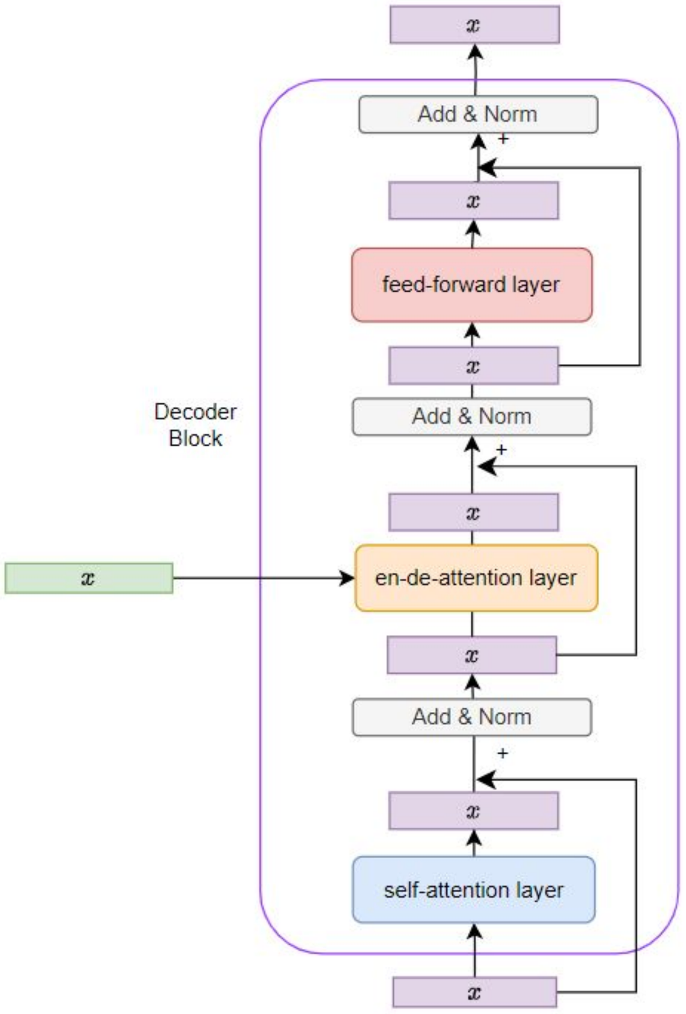


image source

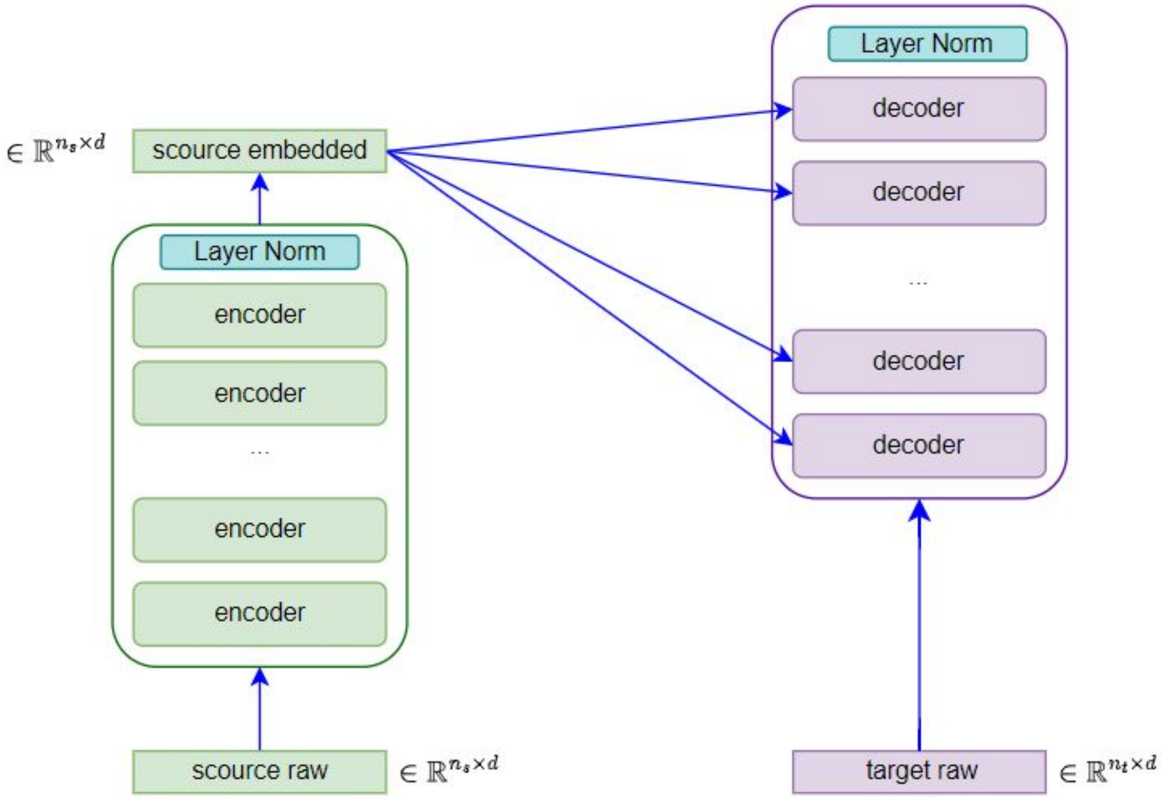
The Decoder Block



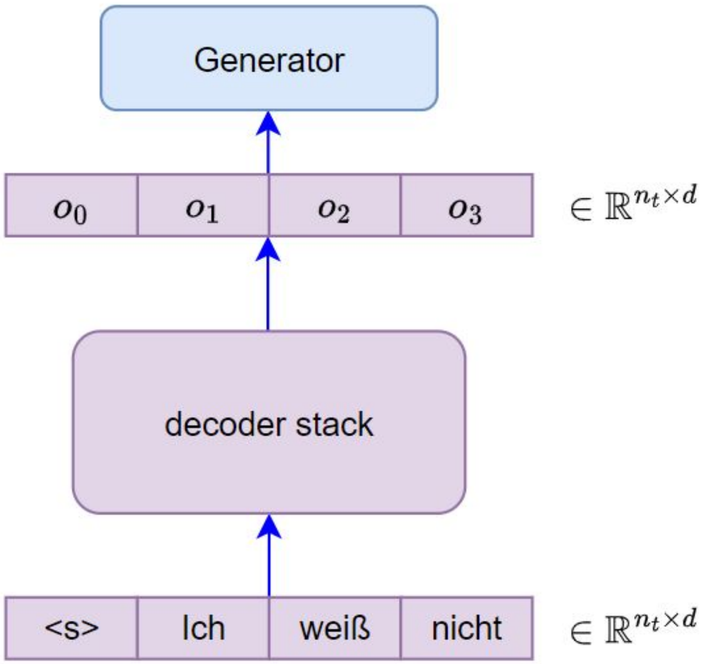
In original paper



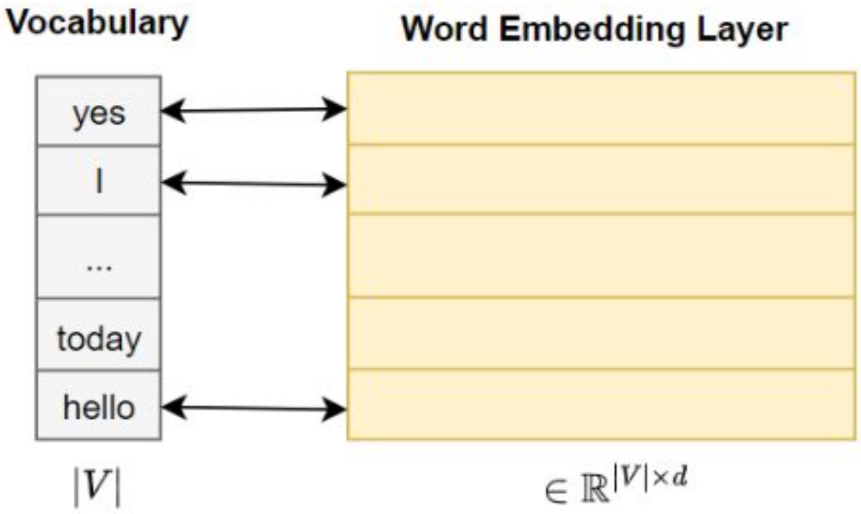
The Decoder Block



The Generator

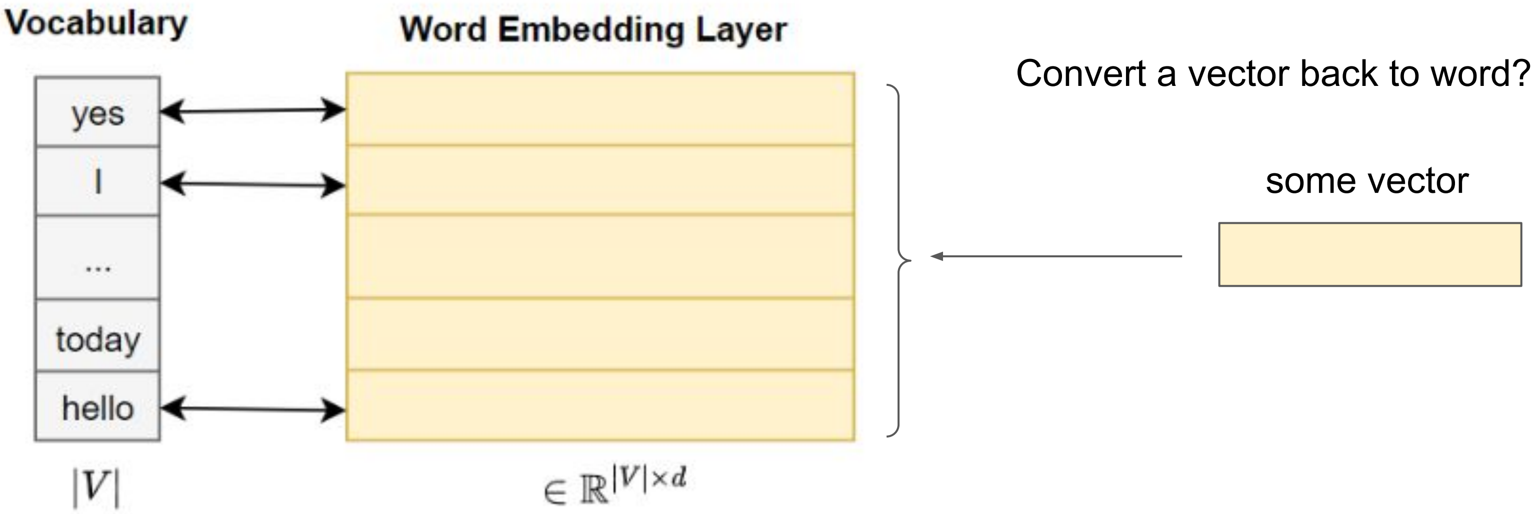


The Generator

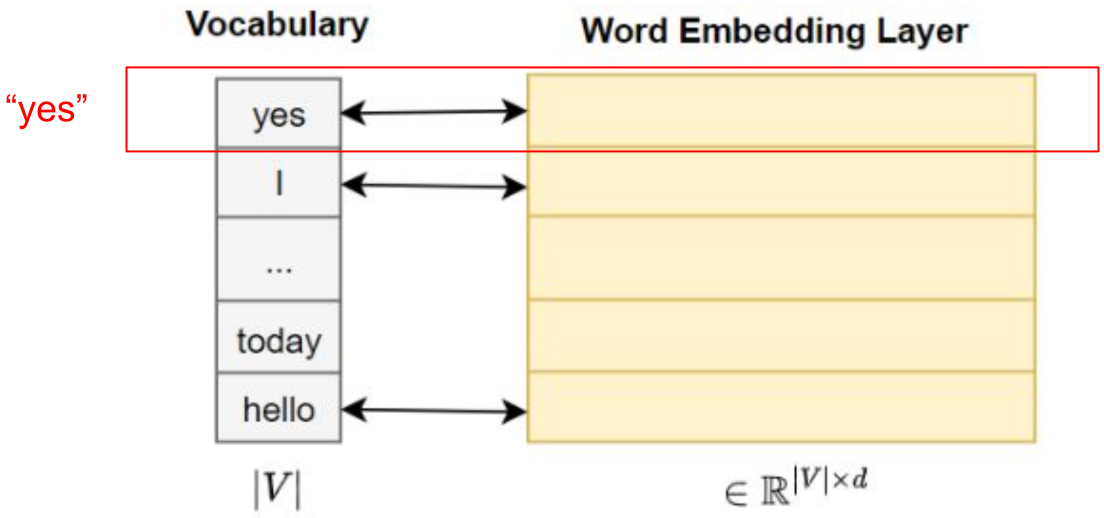


Word embedding: for a fixed word convert to a fixed embedding

The Generator



The Generator



logits $\in \mathbb{R}^{|V|}$

0.9
0.01
0.01
…
0.01

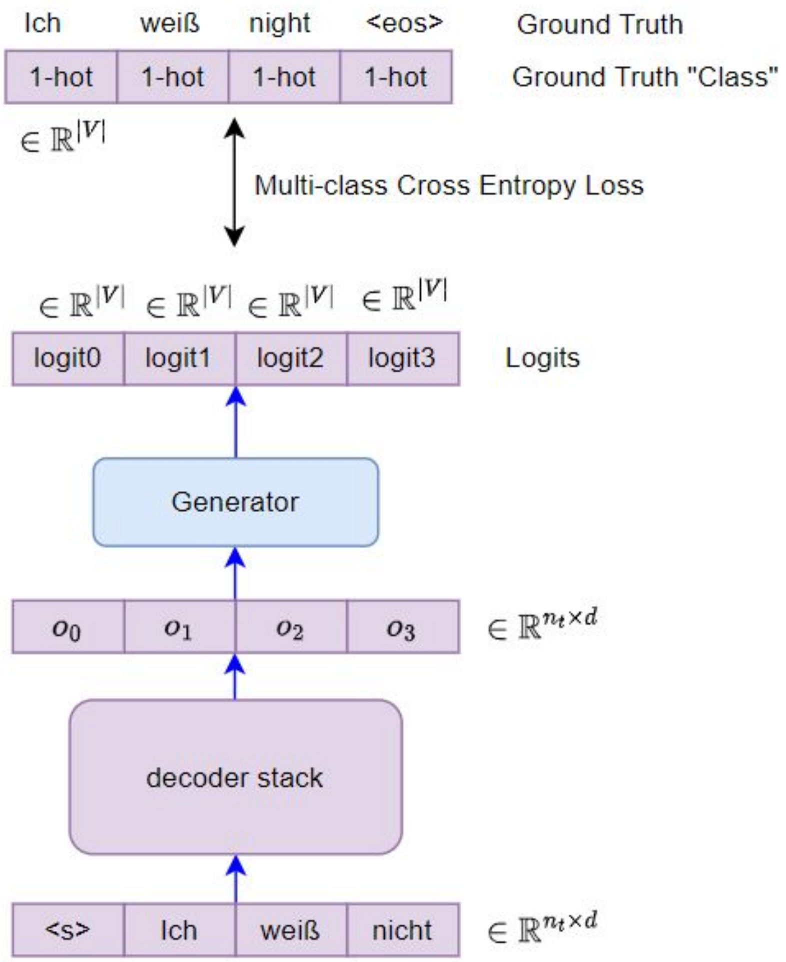
some vector

Which is the most similar word embedding vector?

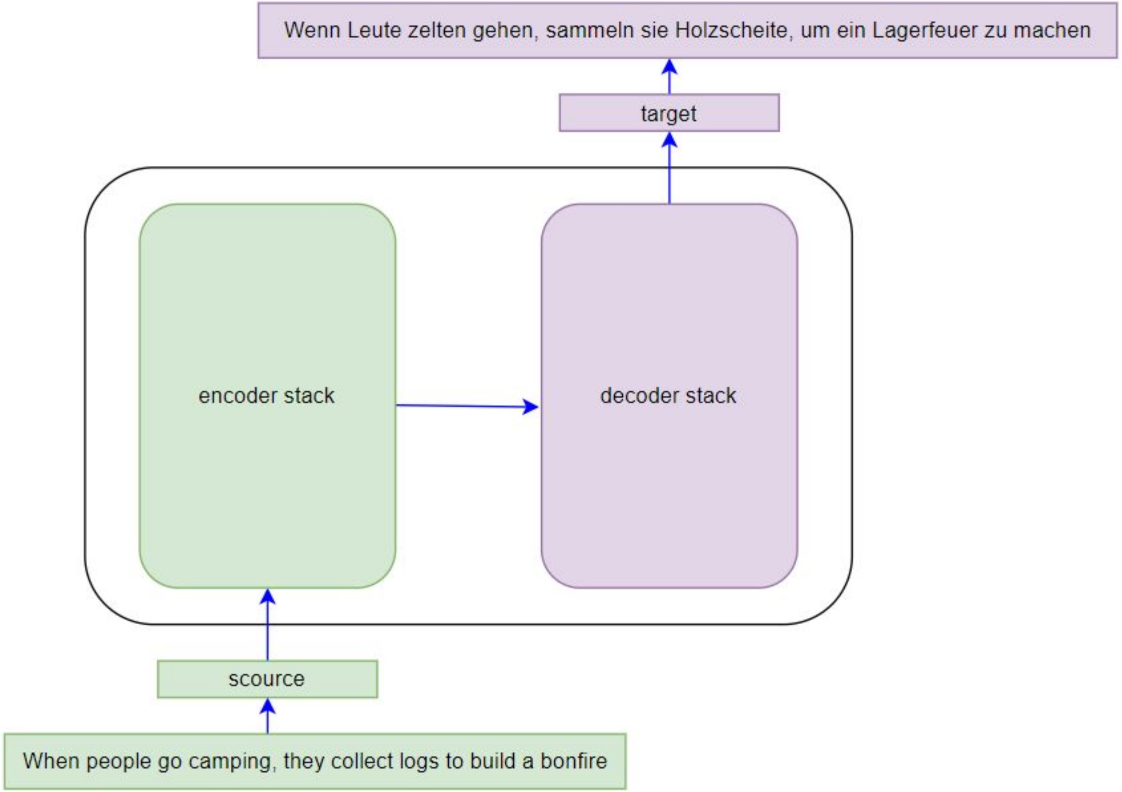
dot product with every word embedding vector
do softmax over result to get logits

The Training Loss

ground truth	**logit**
1	0.9
0	0.01
0	0.01
…	…
0	0.01



The Transformer Model



References

- The annotated transformer http://nlp.seas.harvard.edu/annotated-transformer
- A Gentle Introduction to Positional Encoding in Transformer Models, Part 1 https://machinelearningmastery.com/a-gentle-introduction-to-positional-encodi
ng-in-transformer-models-part-1/#:~:text=What%20Is%20Positional%20Enco
ding%3F,item's%20position%20in%20transformer%20models.
- Database analogy modified from Dive into Deep Learning CH11.1

References

- Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. In International Conference on Learning Representations.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser and Illia Polosukhin. 2017. Attention is All you Need. In Advances in Neural Information Processing Systems. Curran Associates, Inc..
- Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. 2015. Deep Residual Learning for Image Recognition
- Jimmy Lei Ba, Jamie Ryan Kiros and Geoffrey E. Hinton. 2016. Layer Normalization