الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

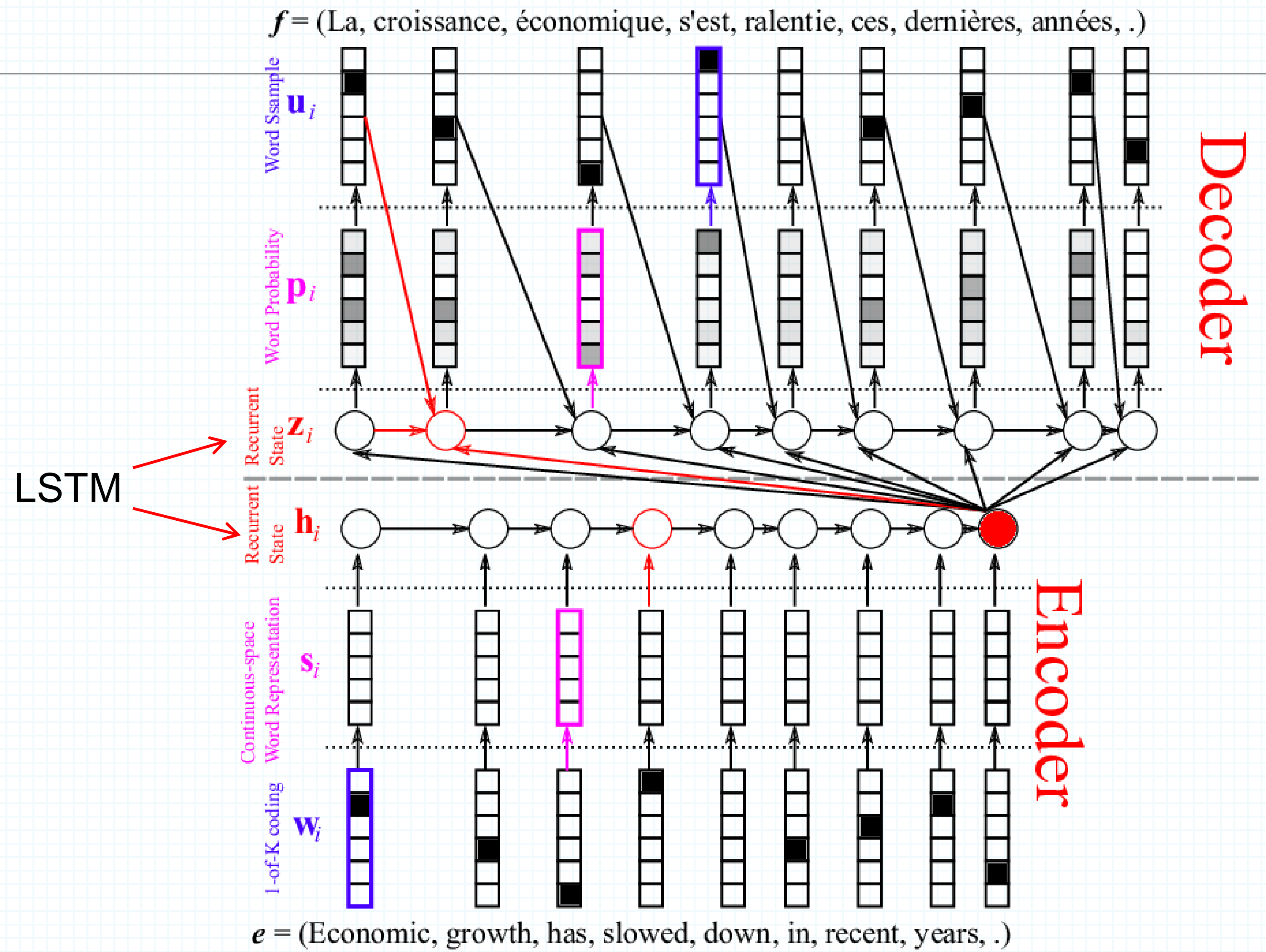
Topic 2 | كلية الهندسة | الذكاء الصنعي العملي

Transformers

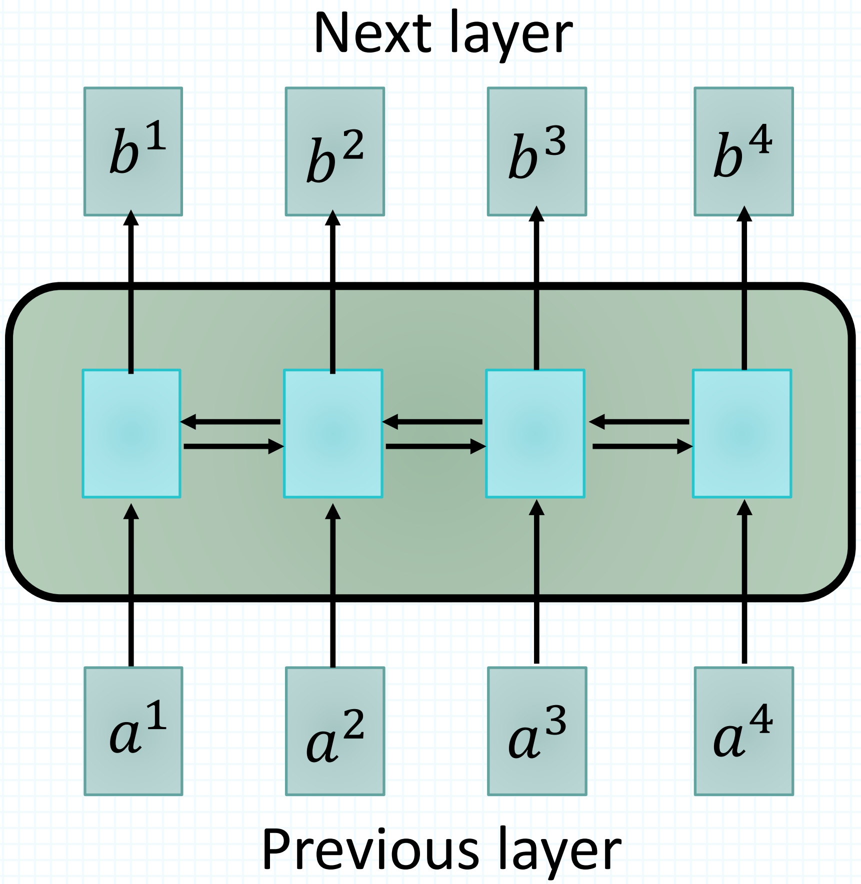
د. رياض سنبل

Transformer
I*n General*

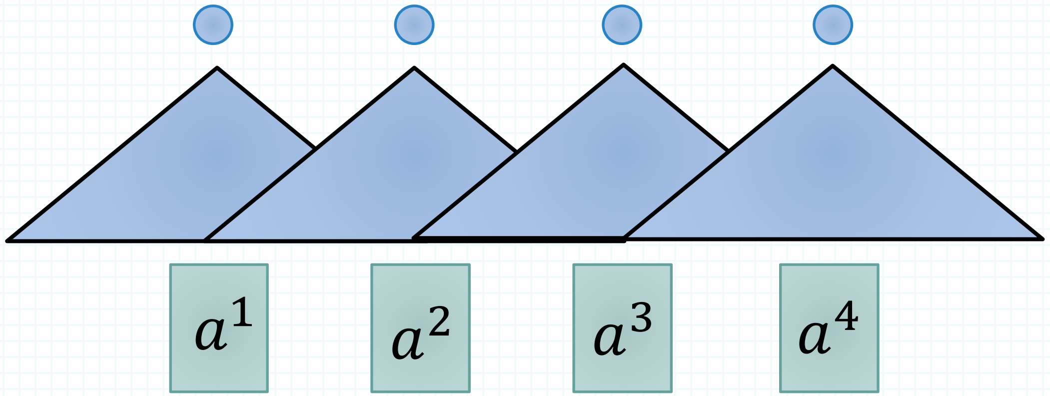
Encoder-Decoder



Sequence



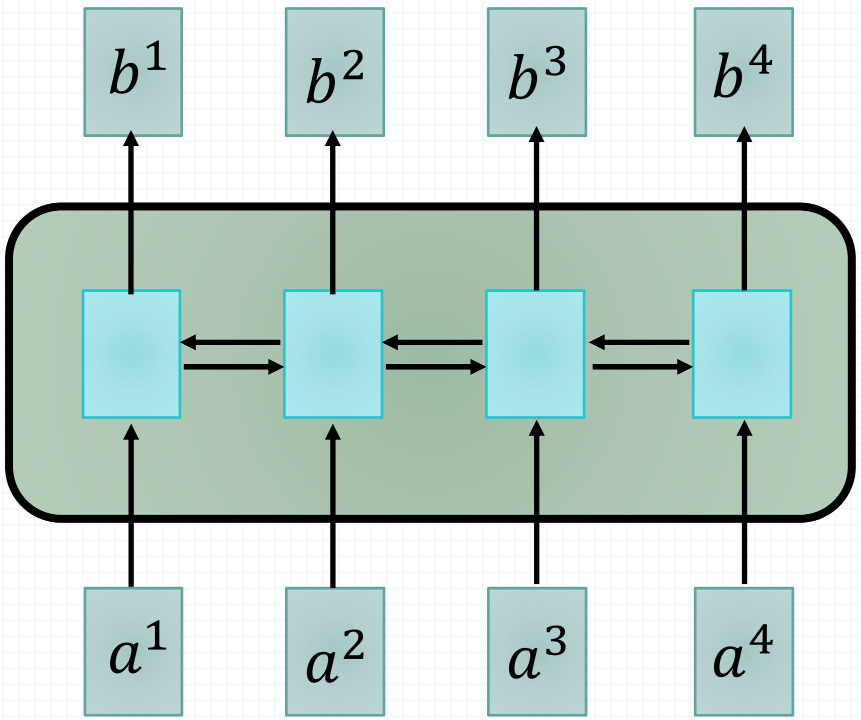
Hard to parallel !



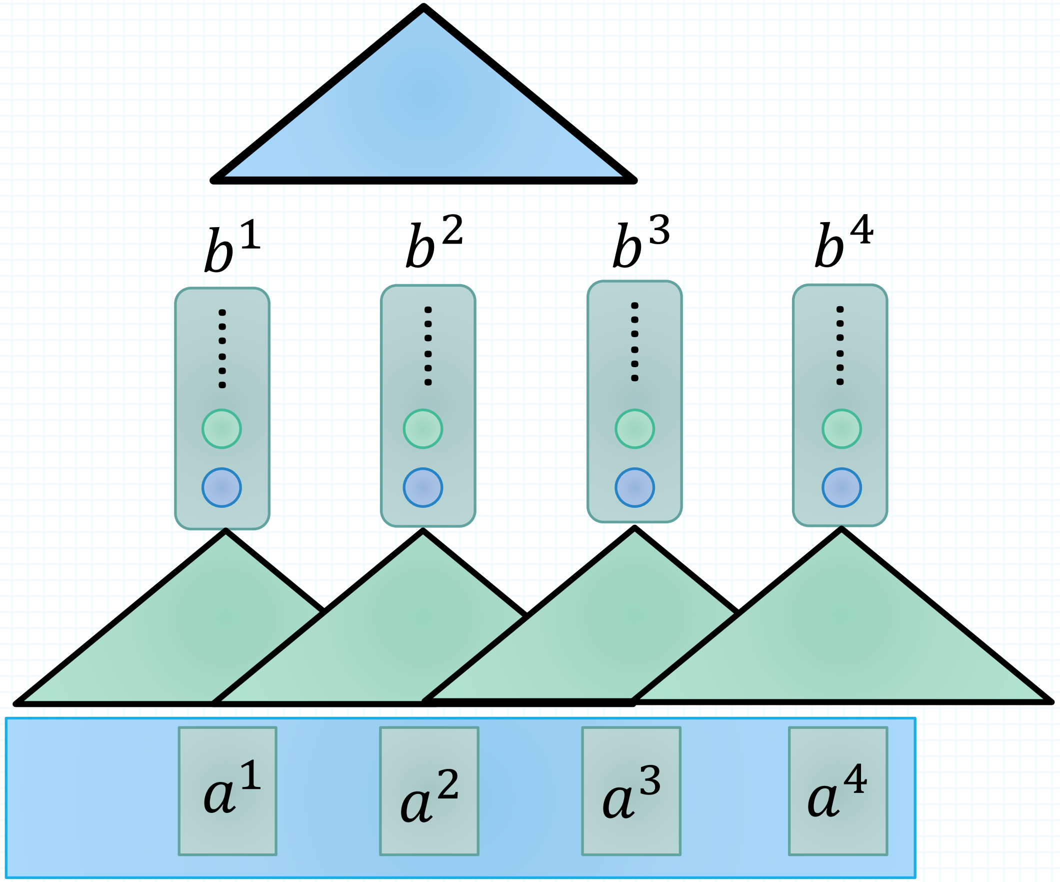
Using CNN to replace RNN

Sequence

Filters in higher layer can consider longer sequence



Hard to parallel

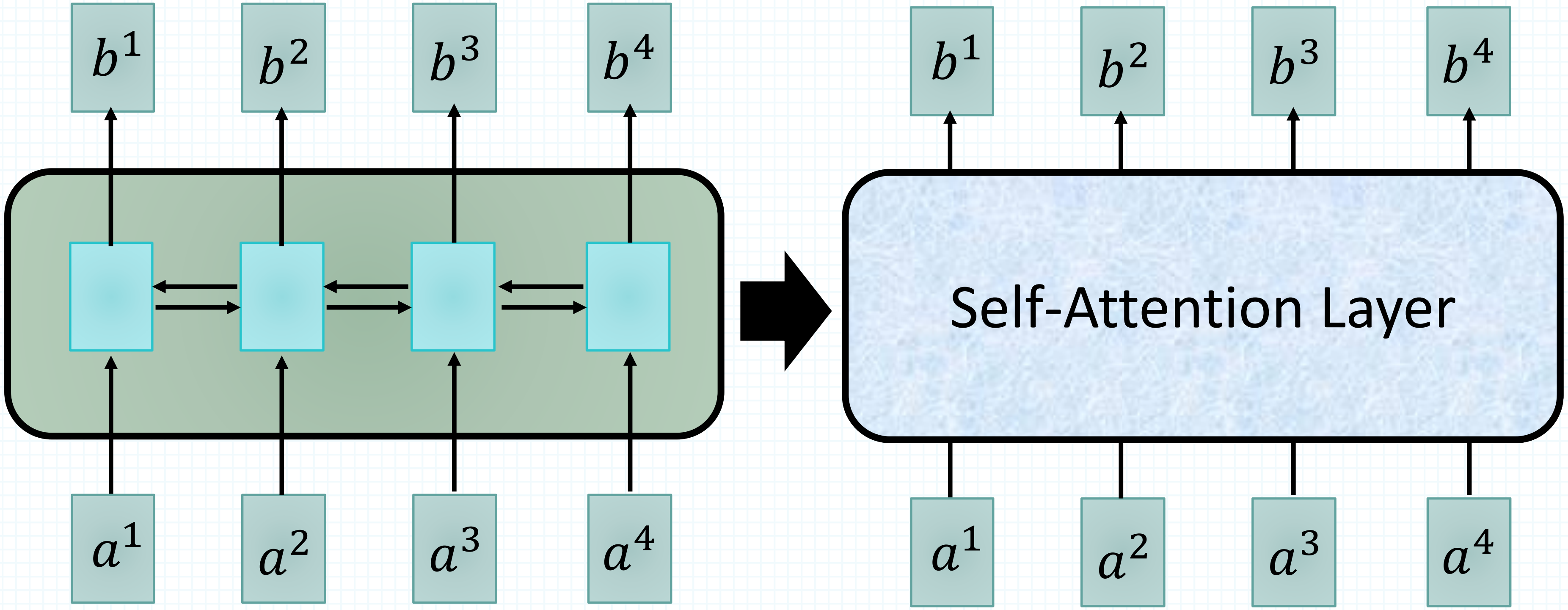


Using CNN to replace RNN (CNN can parallel)

Self-Attention

b^i is obtained based on the whole input sequence.

b^1, b^2, b^3, b^4 can be parallelly computed.



You can try to replace any thing that has been done by RNN with self-attention.

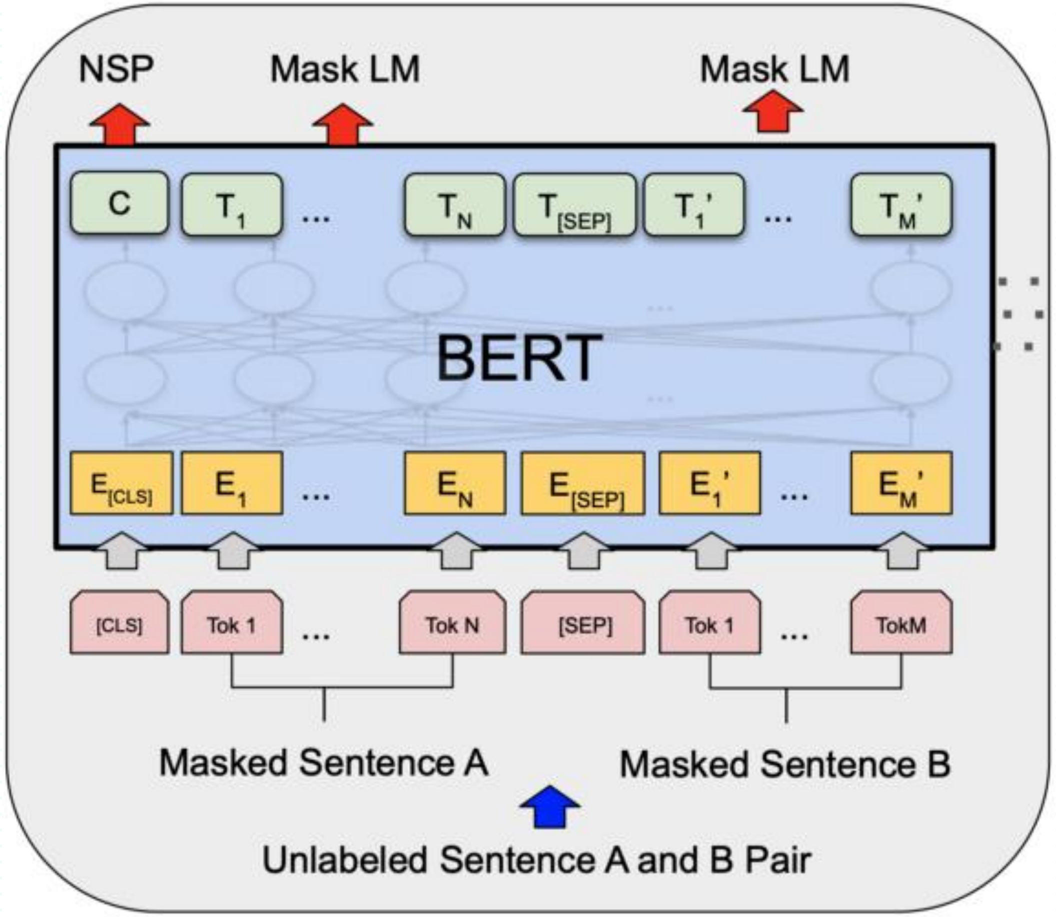
BERT

Next Sentence Prediction

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

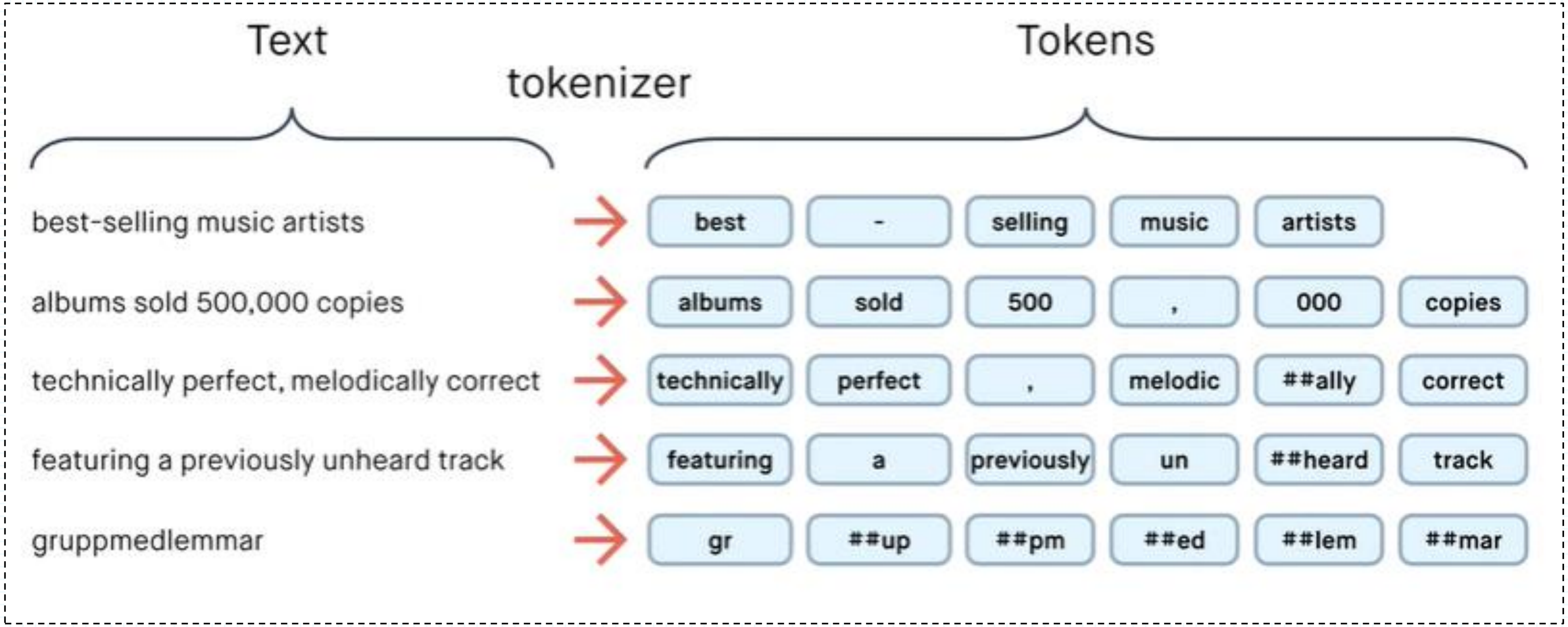
2018

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

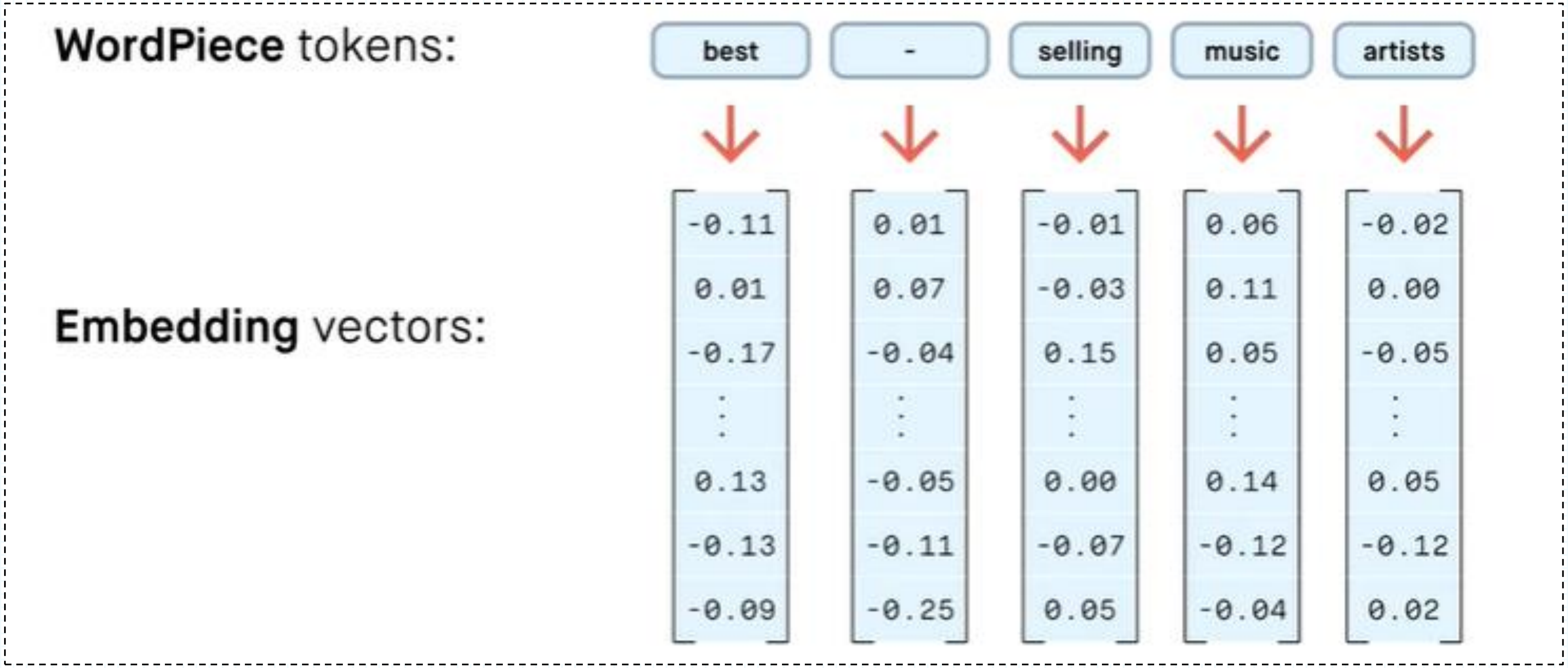


Pre-training

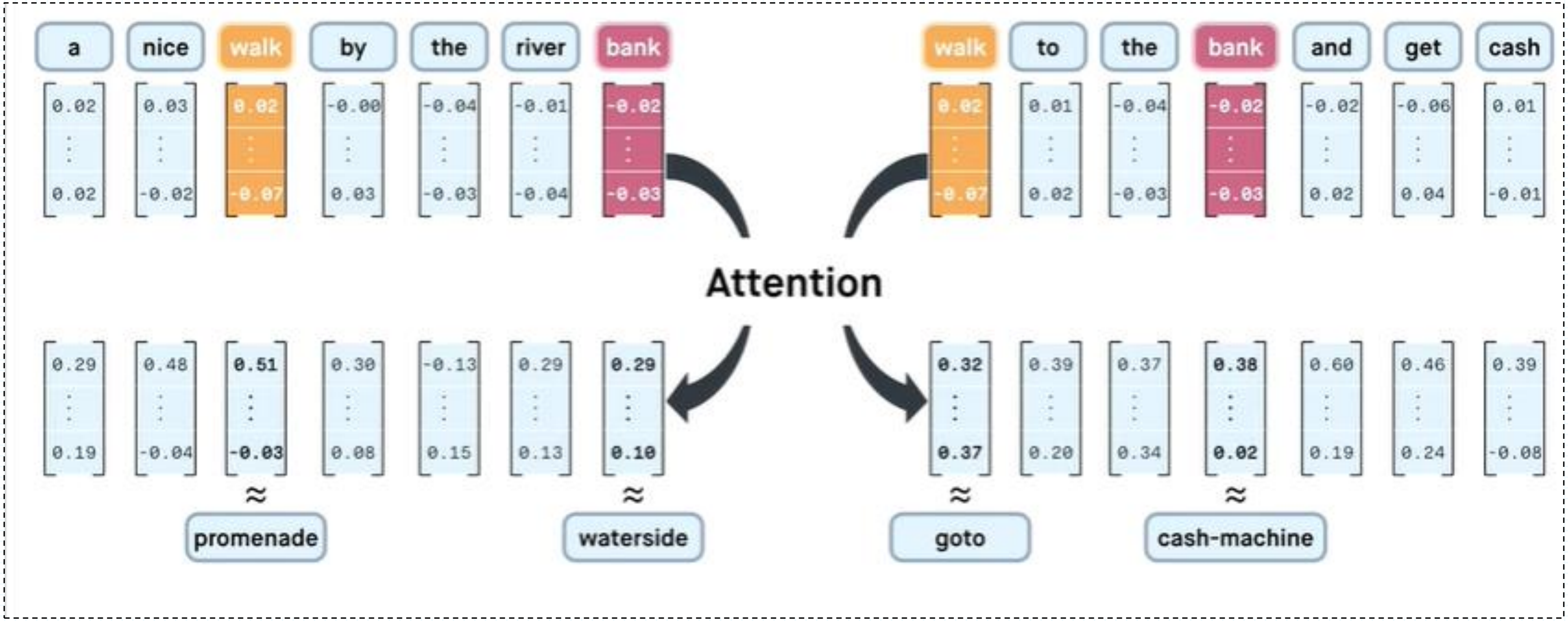
Tokenization

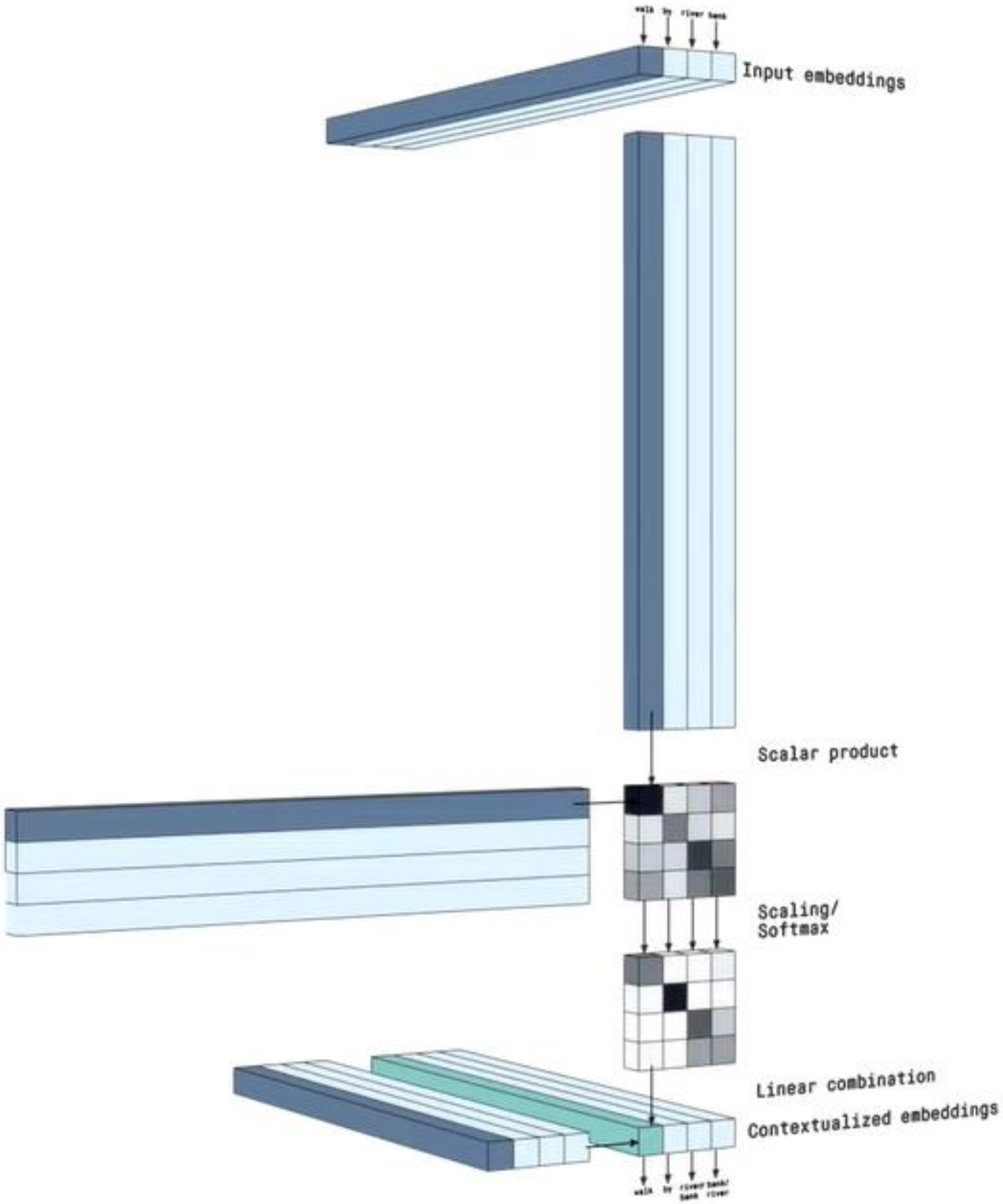


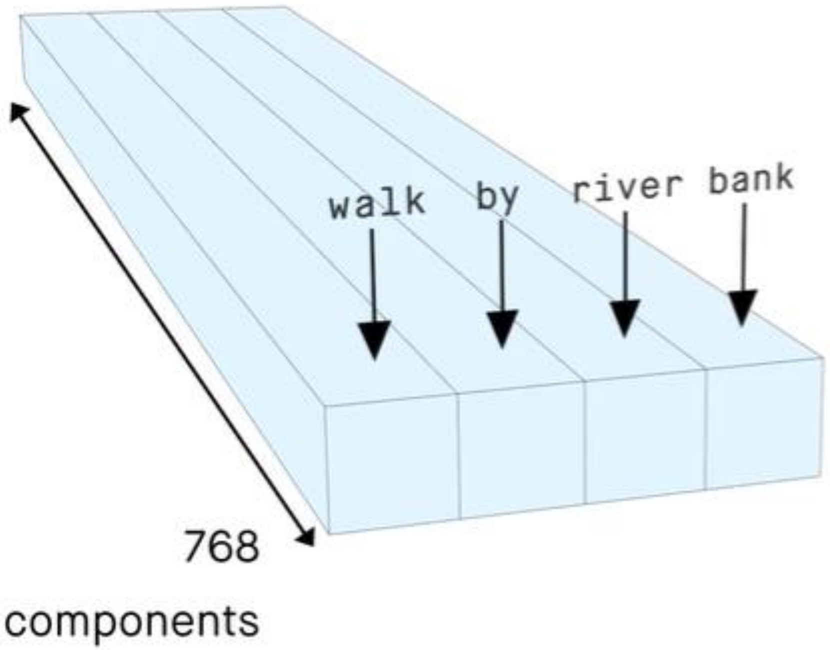
Initial Embedding

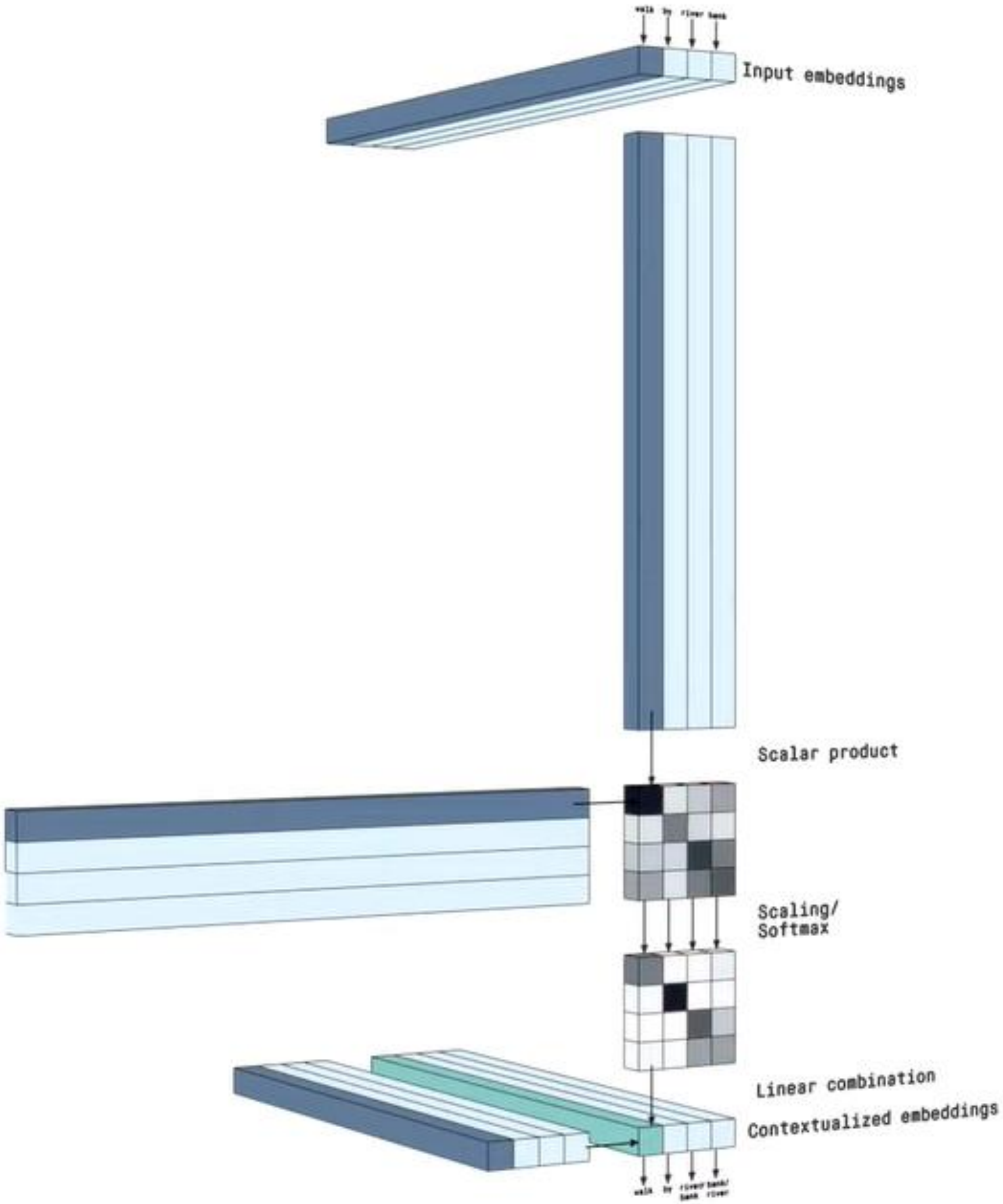


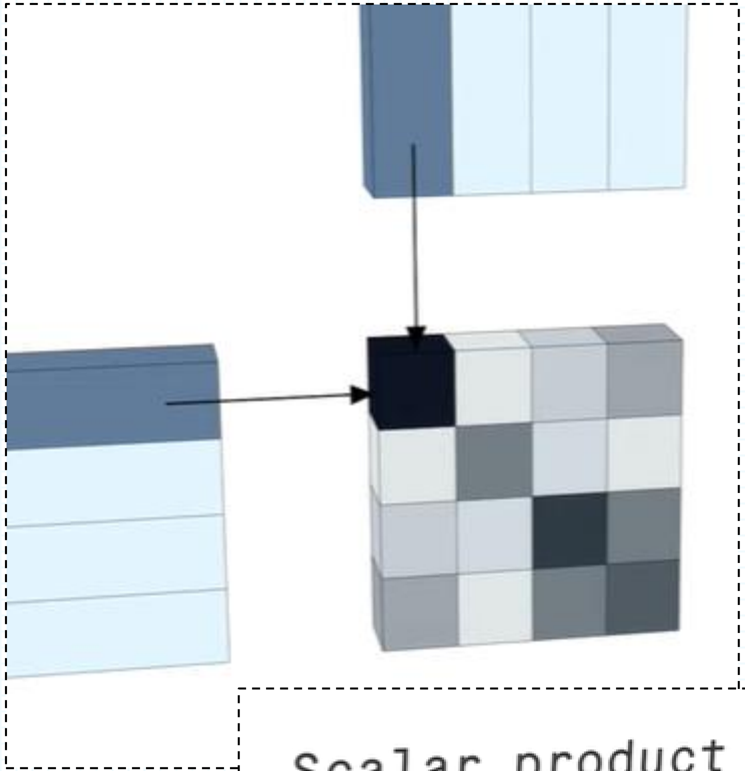
Contextual Embedding



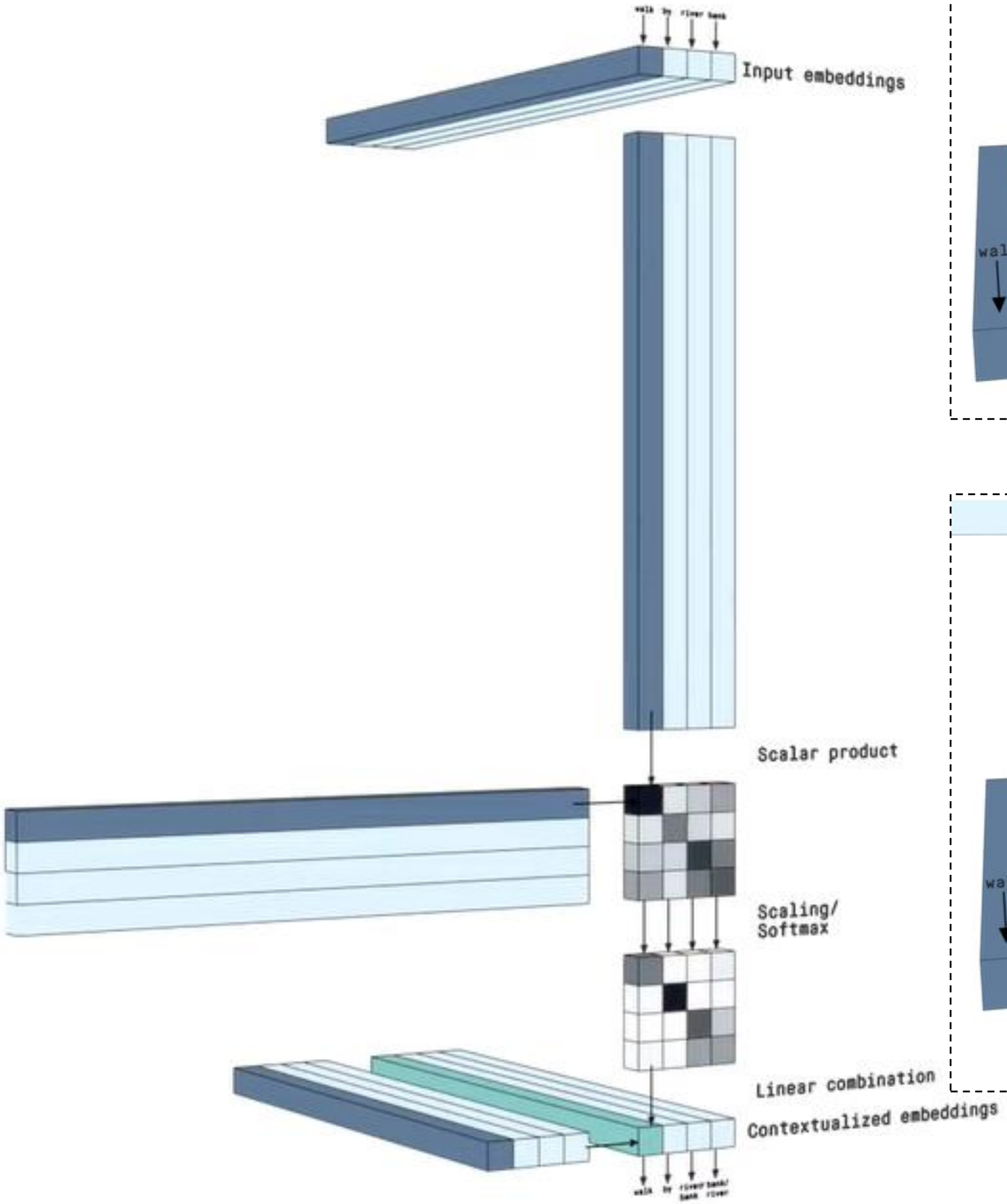

Input embeddings
Scalar product
Scaling/
Softmax
Linear combination
Contextualized embeddings

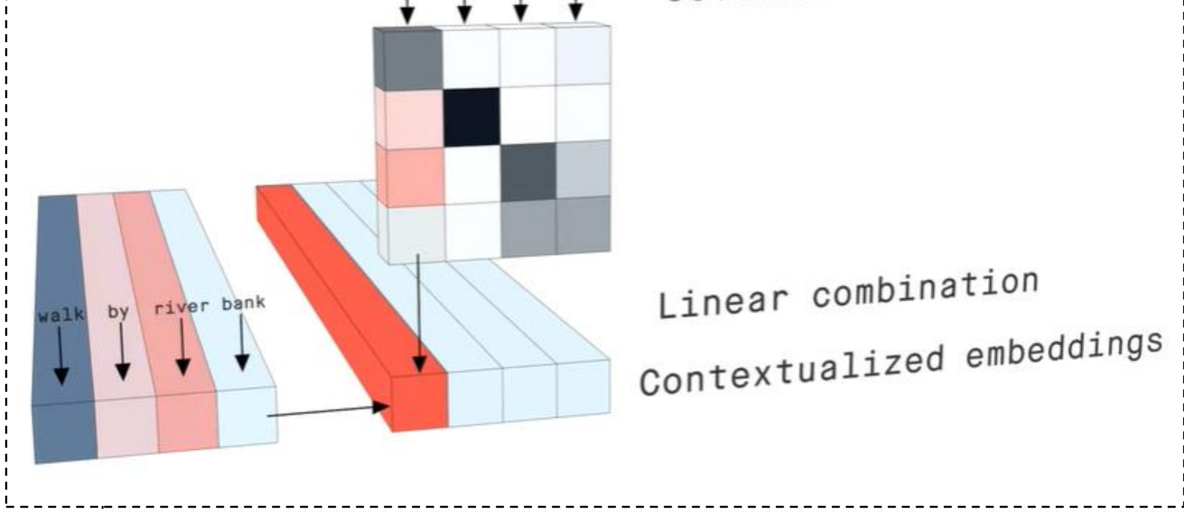

walk
by
river
bank
768
components

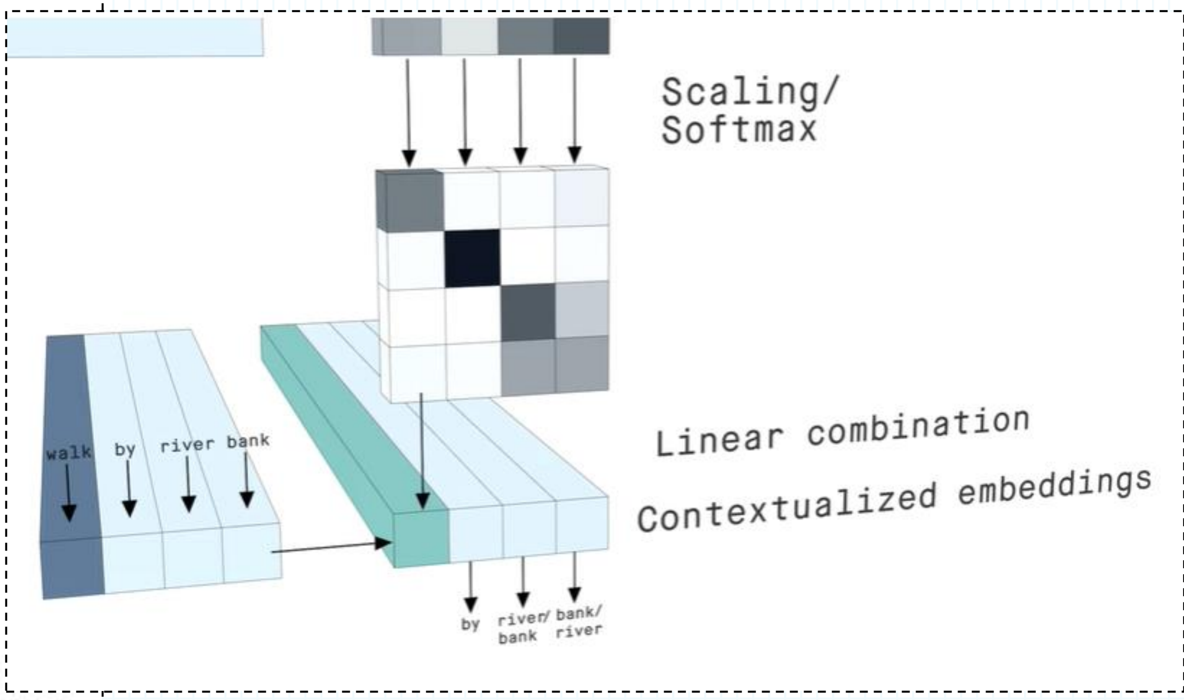




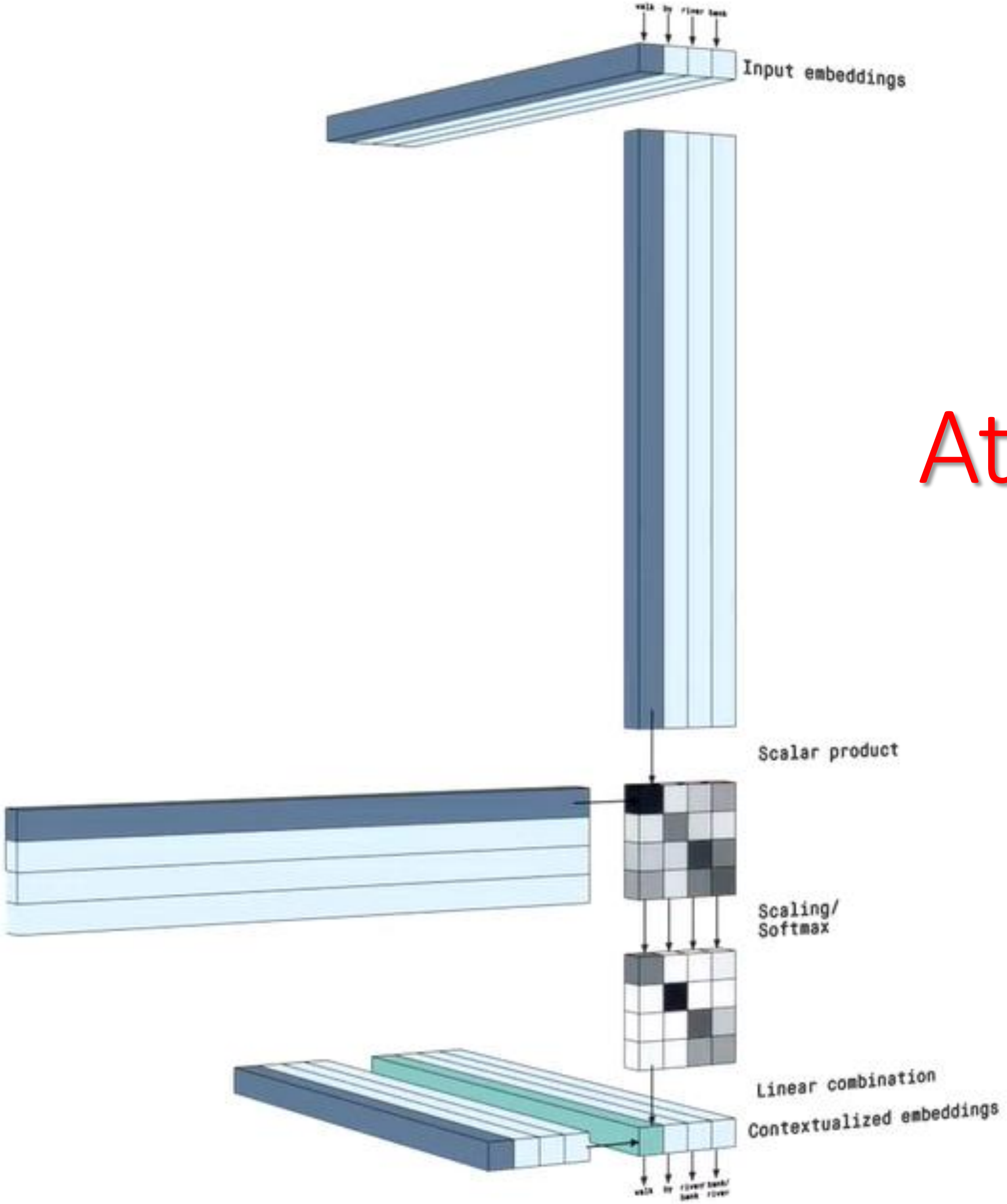
$$\frac{\text{Scalar product}}{\sqrt{\text{vector size}}}$$

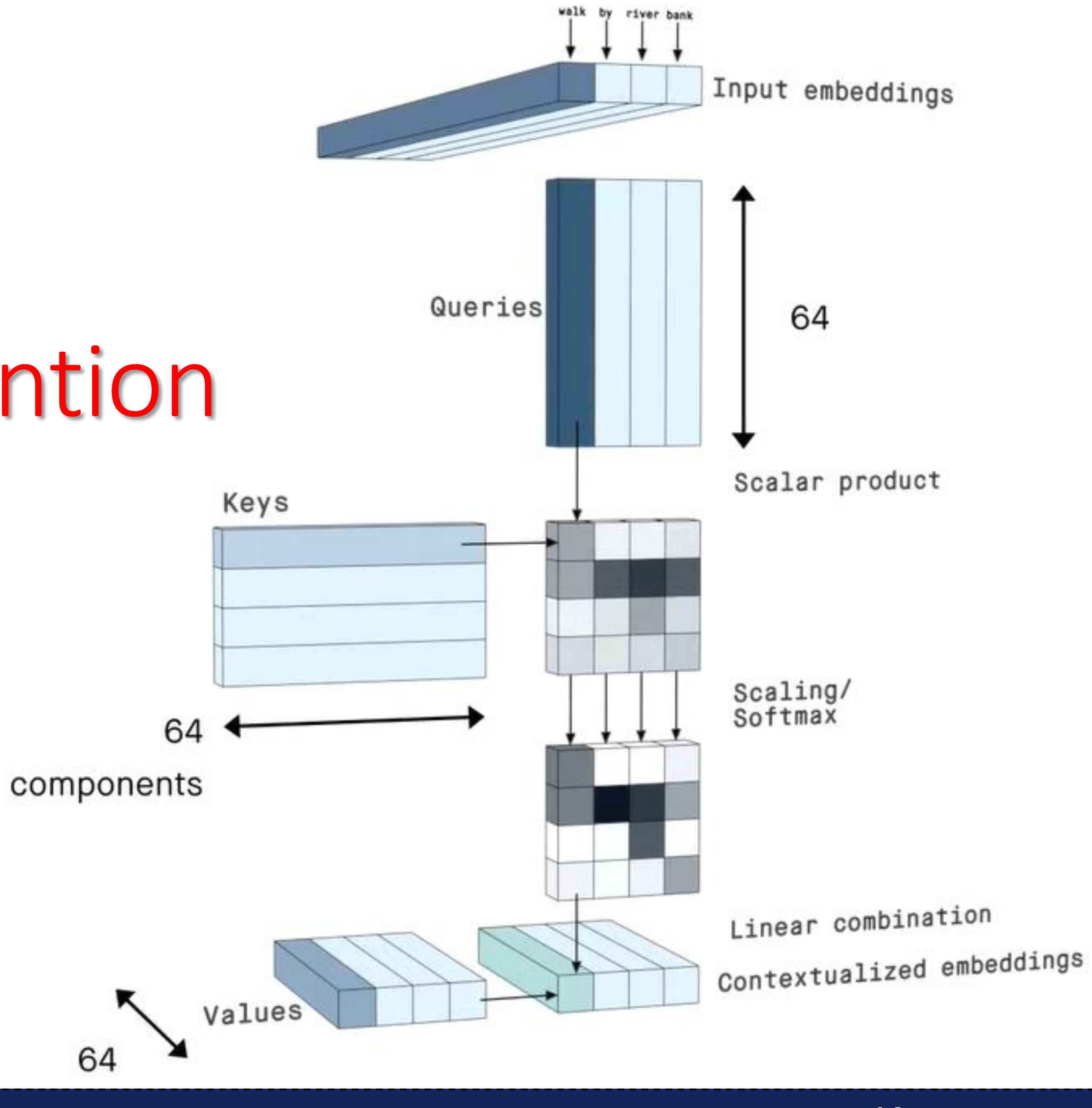

Input embeddings
Scalar product
Scaling/
Softmax
Linear combination
Contextualized embeddings


walk
by
river
bank
Linear combination
Contextualized embeddings

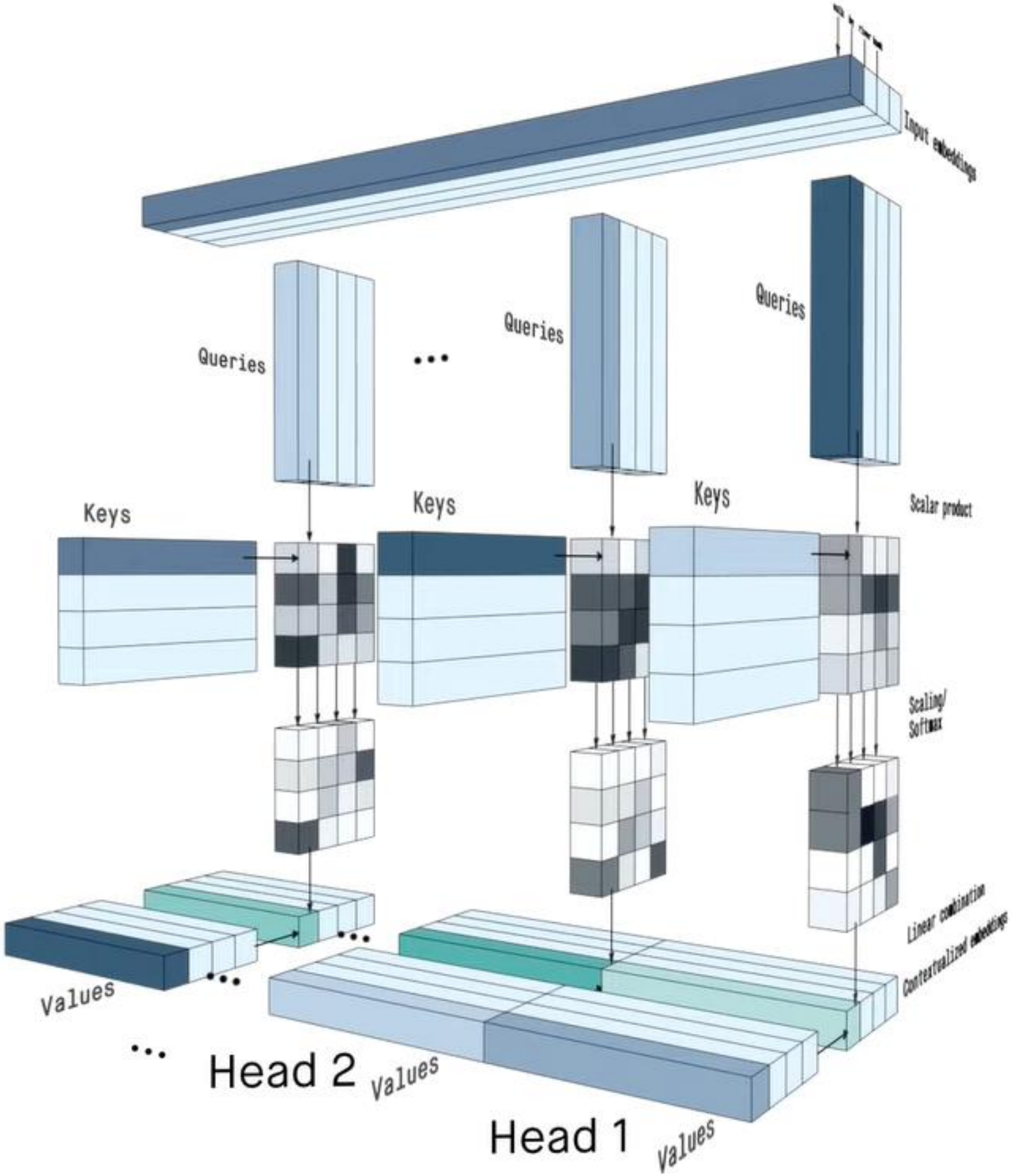

Scaling/
Softmax
walk
by
river
bank
Linear combination
Contextualized embeddings
by
river/
bank
bank/
river

Attention

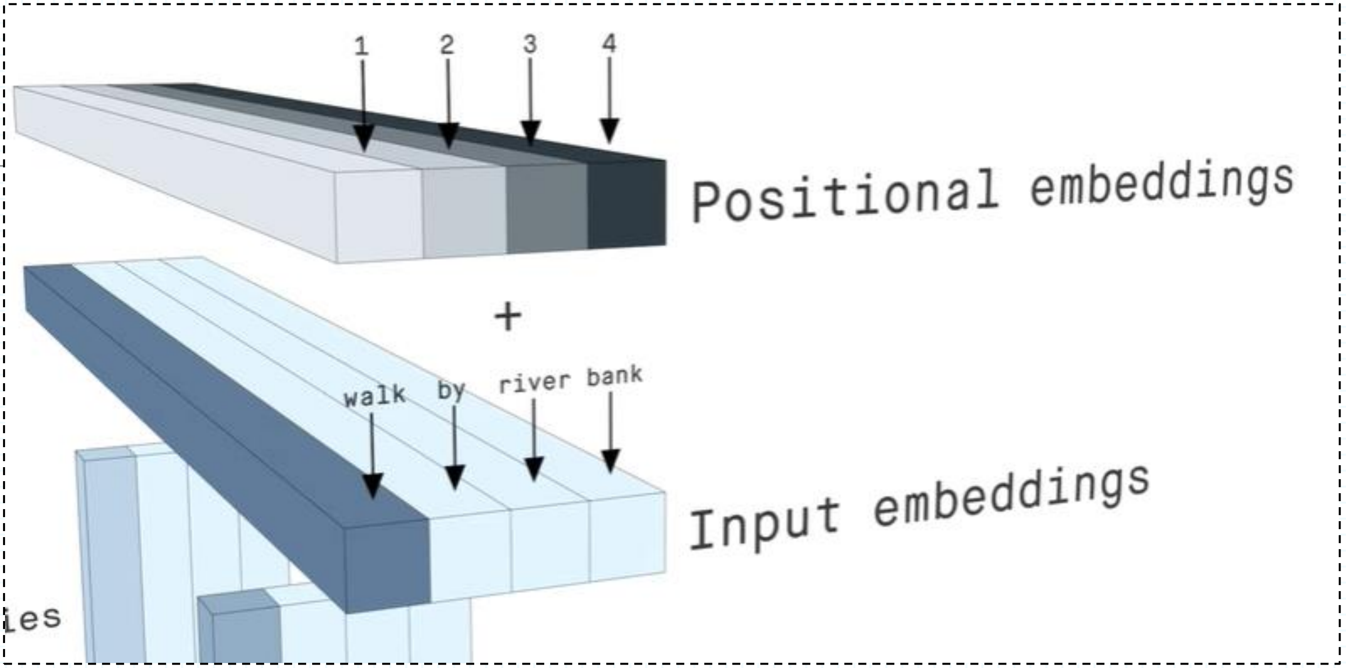

walk by river bank
Input embeddings
Scalar product
Scaling/
Softmax
Linear combination
Contextualized embeddings


walk by river bank
Input embeddings
Queries
64
Scalar product
Keys
Scaling/
Softmax
64
components
Linear combination
Contextualized embeddings
Values
64

Multi-head attention



Enrich the input!



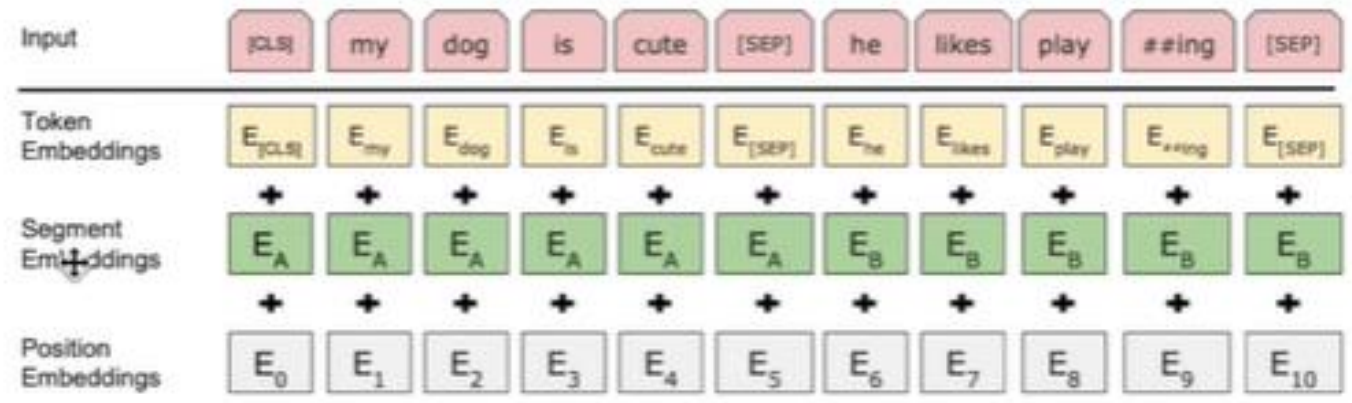
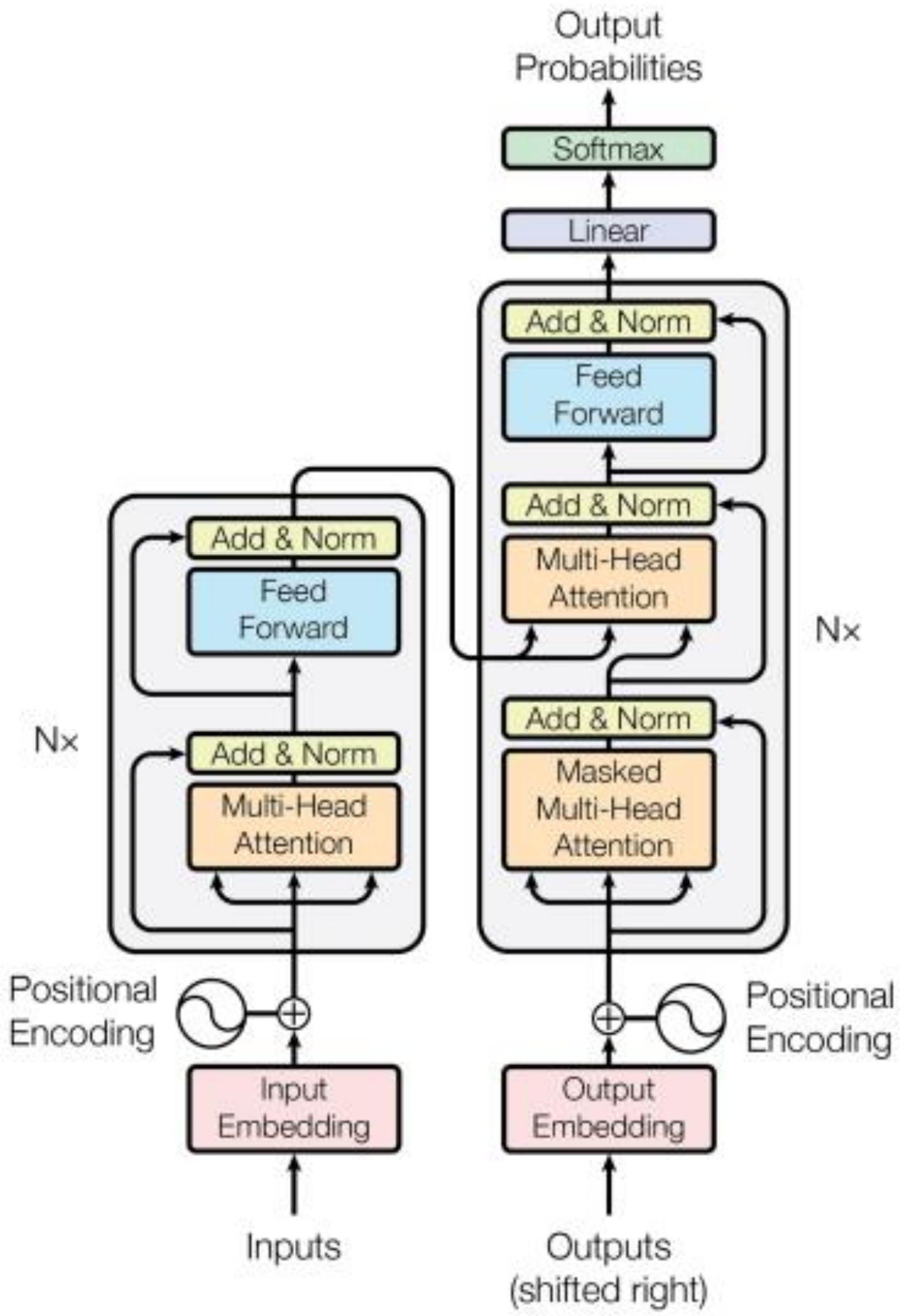


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

So.. What are Transformers in general!

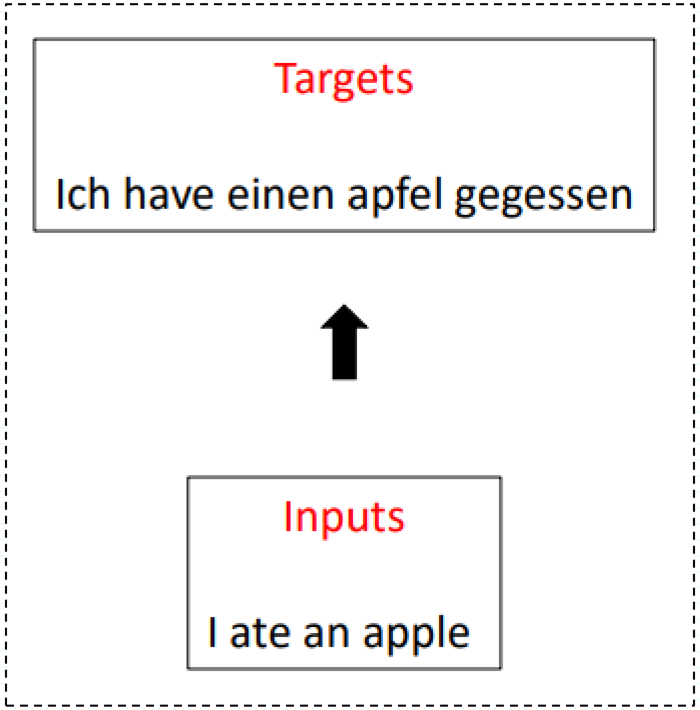
Transformers

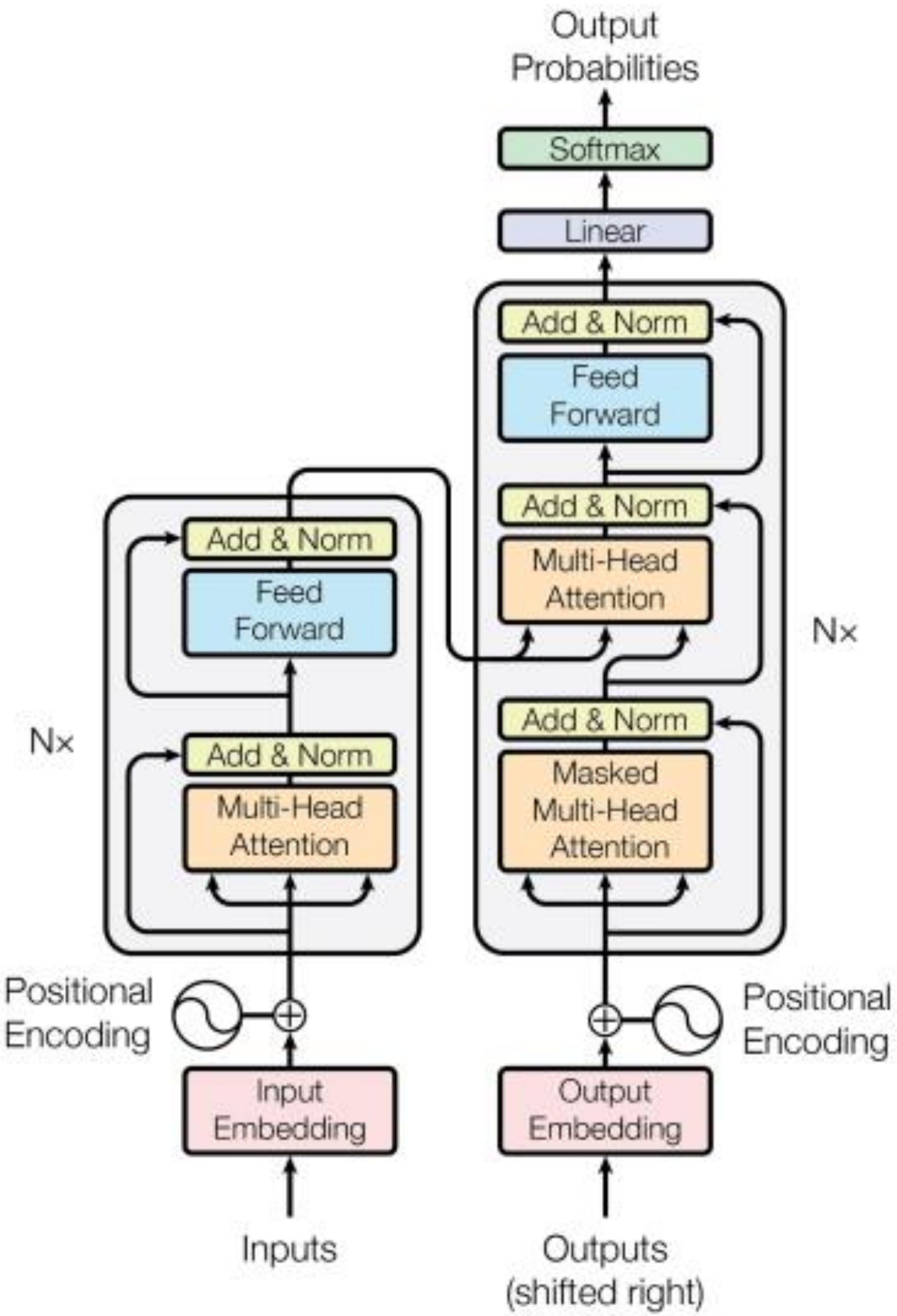
- Transformers are a type of neural network architecture that transforms or changes an input sequence into an output sequence.
- They do this by learning context and tracking relationships between sequence components.
- And break the problem into two parts:
  - An encoder (e.g., Bert)
  - A decoder (e.g., GPT)



Transformers

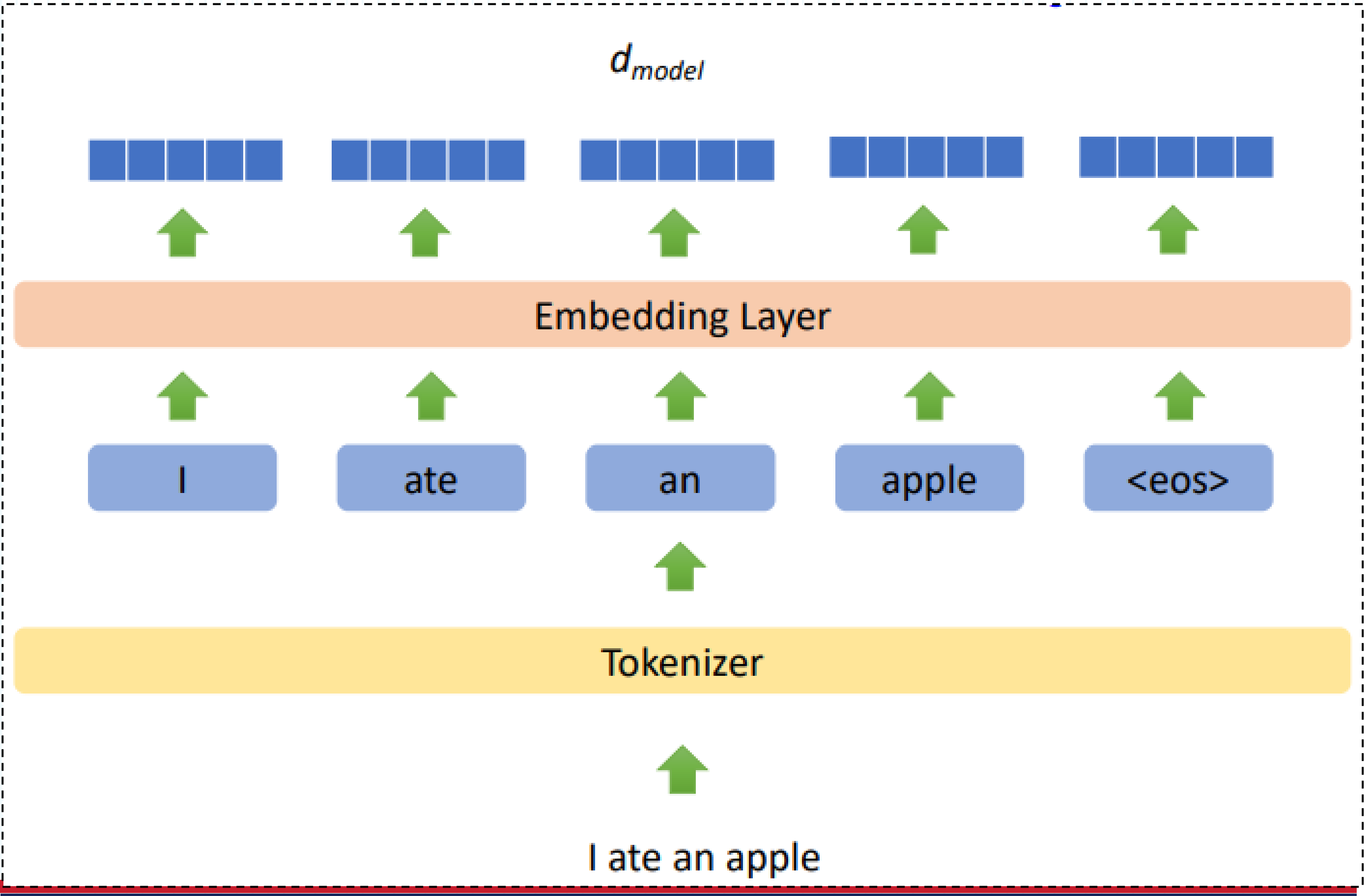
- Example: Machine Translation

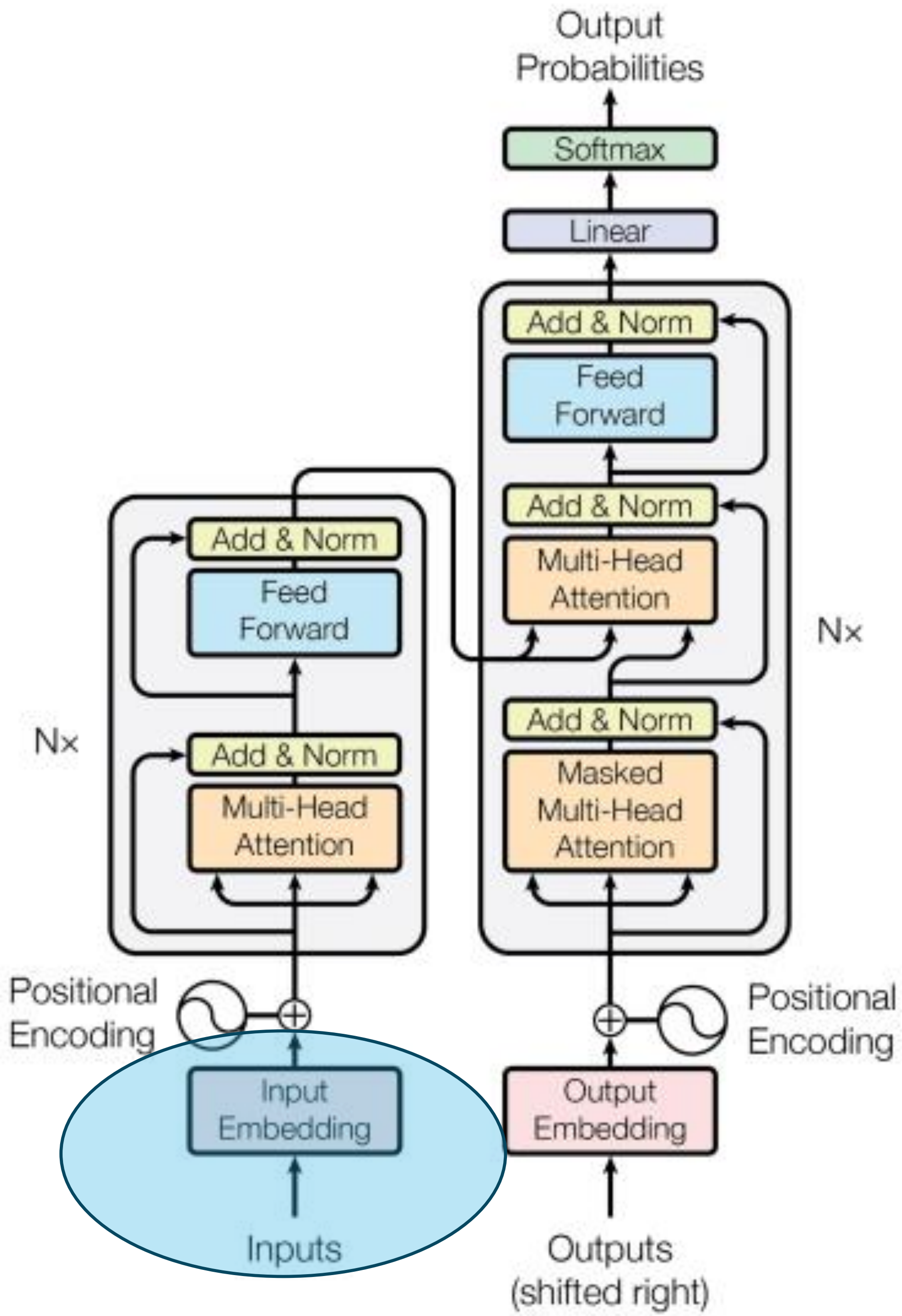




Transformers

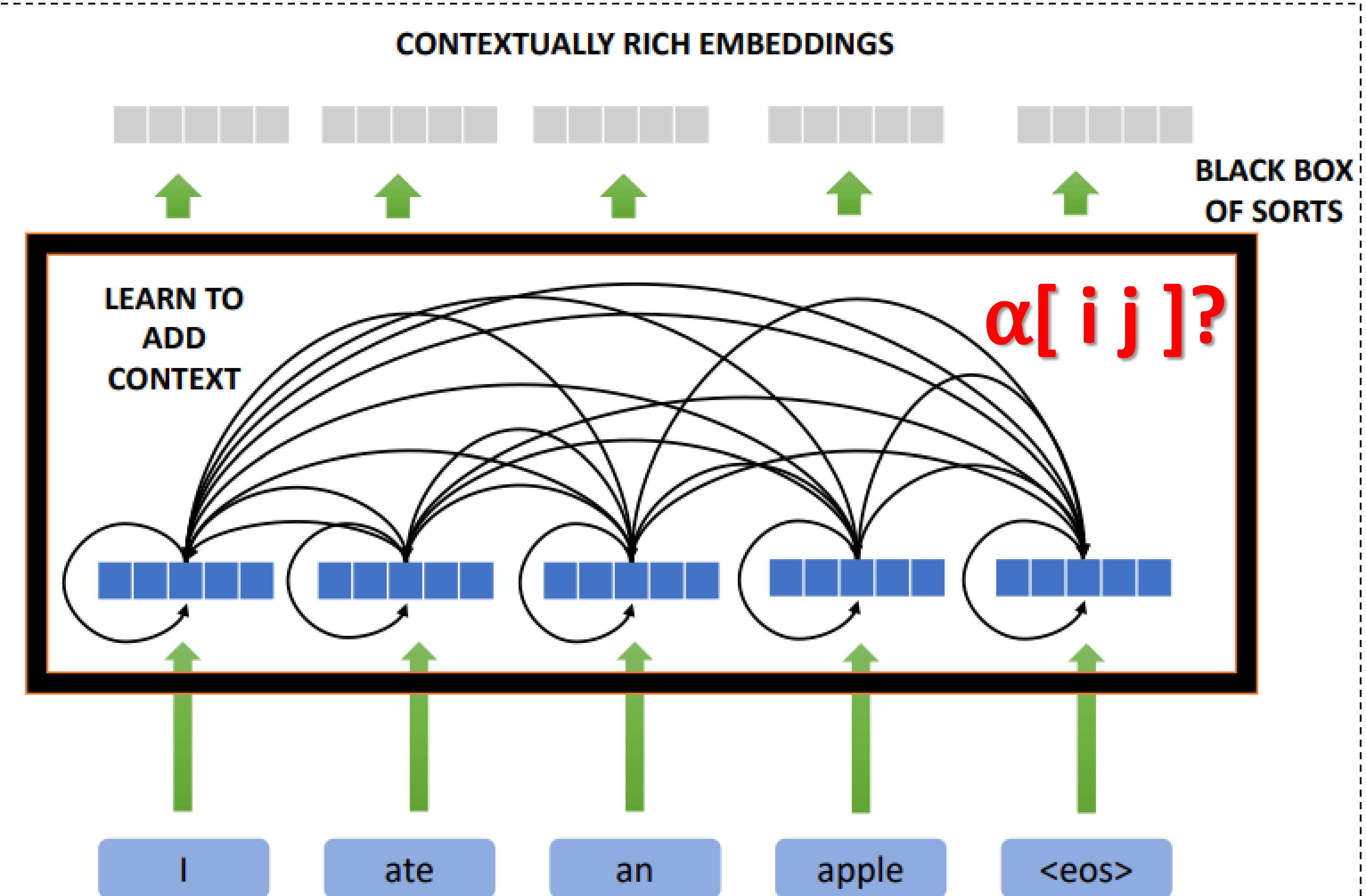
- Processing Input

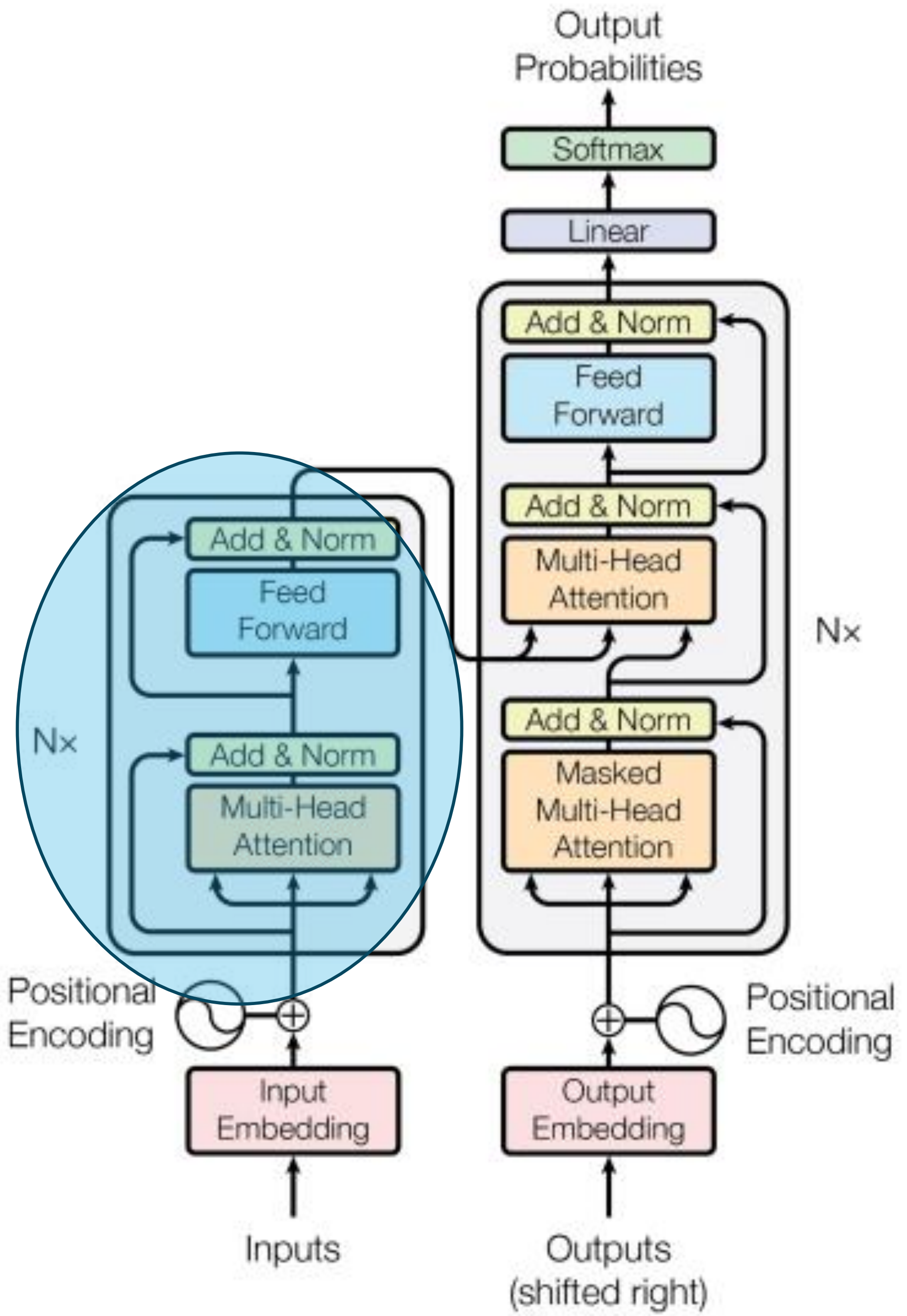




Transformers

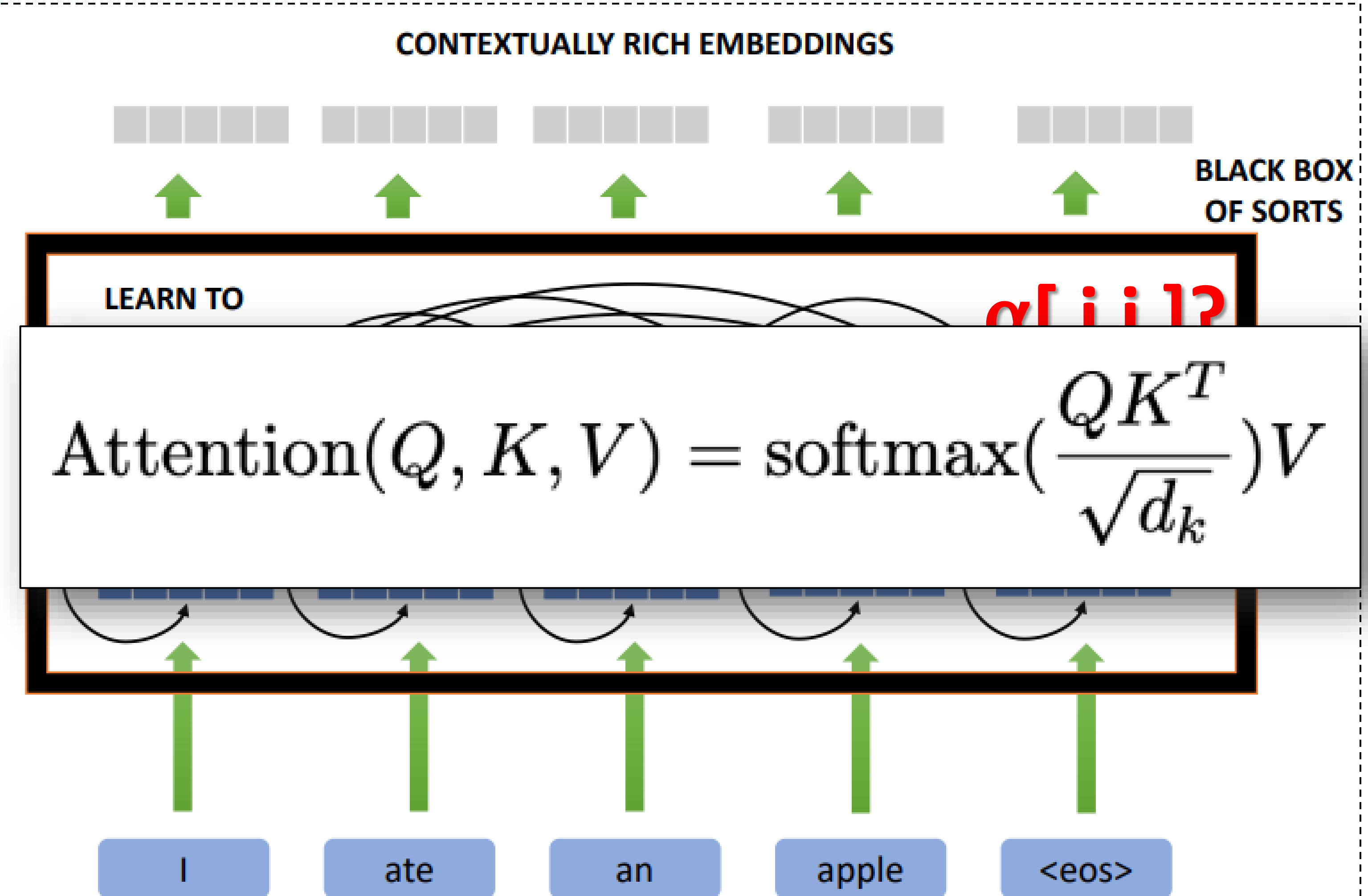
- Learn to add context



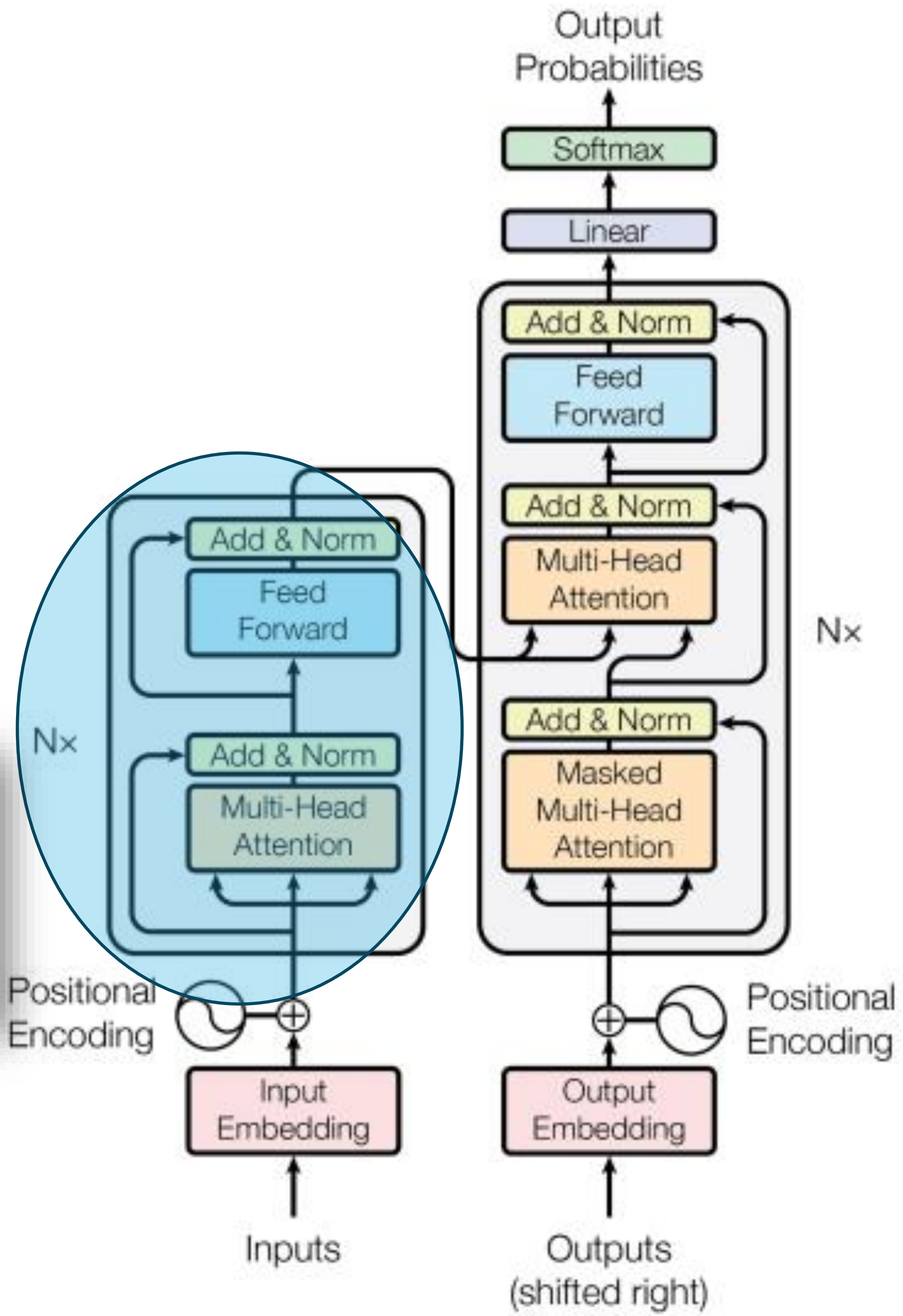


Transformers

- Learn to add context



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



Attention Concepts

- In the attention mechanism, we can draw an analogy to Information Retrieval (IR).

{Query: “Order details of order_104”}

OR

{Query: “Order details of order_106”}

A **query** (an embedding vector representing what we want to find more about — e.g., a specific token or position in a sequence)

A set of **keys** (representing the indexed or stored information — i.e., all other tokens in the context)

{Key, Value store}


```
{"order_100": {"items":"a1", "delivery_date":"a2", ...}},
{"order_101": {"items":"b1", "delivery_date":"b2", ...}},
{"order_102": {"items":"c1", "delivery_date":"c2", ...}},
{"order_103": {"items":"d1", "delivery_date":"d2", ...}},
{"order_104": {"items":"e1", "delivery_date":"e2", ...}},
{"order_105": {"items":"f1", "delivery_date":"f2", ...}},
{"order_106": {"items":"g1", "delivery_date":"g2", ...}},
{"order_107": {"items":"h1", "delivery_date":"h2", ...}},
{"order_108": {"items":"i1", "delivery_date":"i2", ...}},
{"order_109": {"items":"j1", "delivery_date":"j2", ...}},
{"order_110": {"items":"k1", "delivery_date":"k2", ...}}
```


And a set of **values** (the actual content or information associated with each key).

Attention Concepts

A set of **keys** (representing the ind ... ed

- In the attention mechanis
an analogy to Information

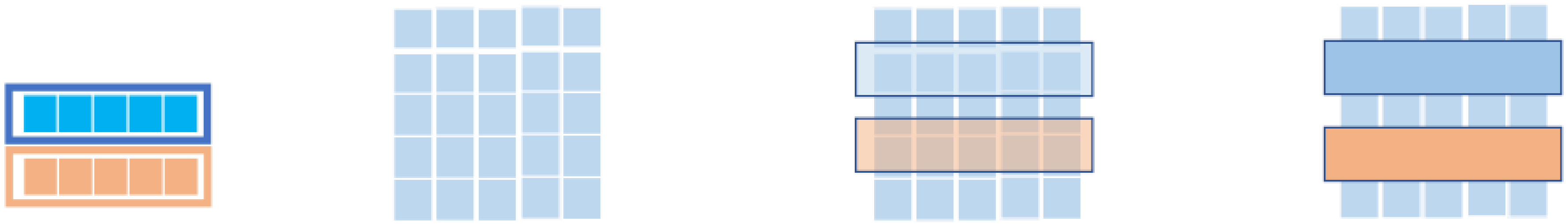
Attention allows the model to dynamically retrieve relevant information (values) from the context, based on how similar other tokens (keys) are to the current focus (query).

The attention process works as follows:

1. We compute the similarity between the query and each key (usually using a dot product).
2. This gives us a set of attention scores, indicating how much focus we should give to each position in the sequence.
3. We then apply a softmax to normalize the scores into a probability distribution.
4. Finally, we compute a weighted sum of the values based on these scores to produce the attention output.

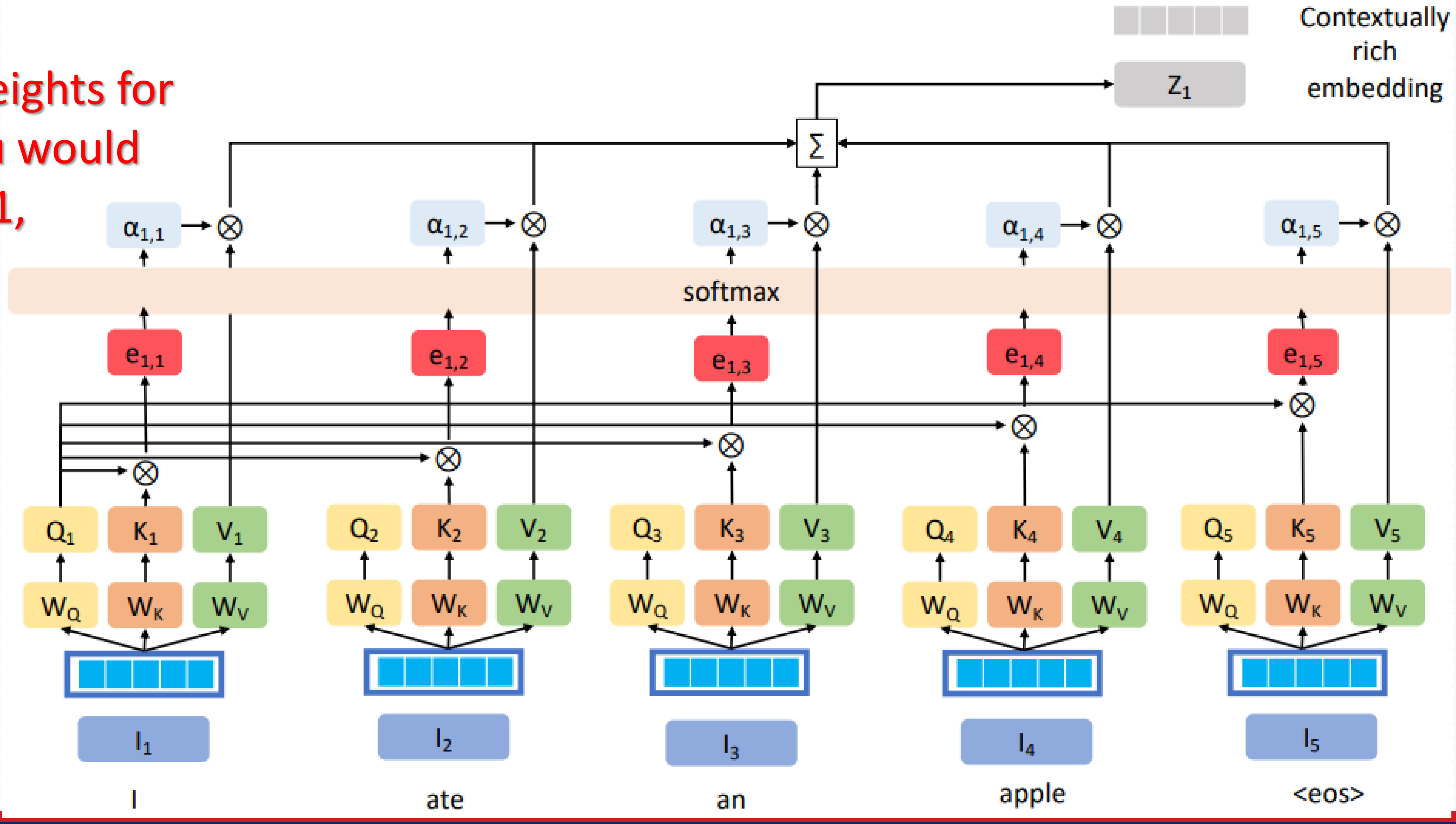
Attention

- More formal!



Attention

To calculate attention weights for input I1, you would use query q1, and all keys

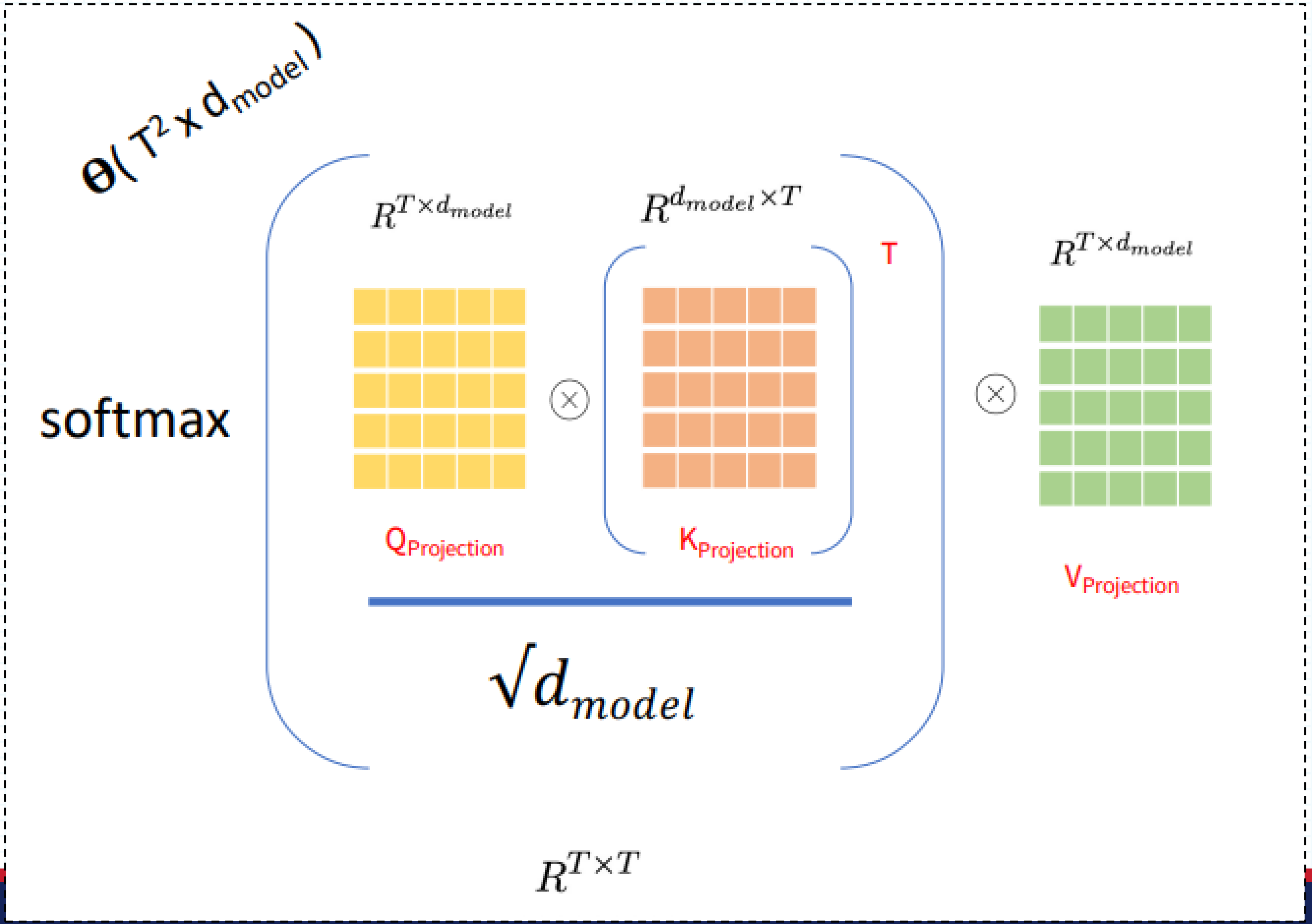


Of I1

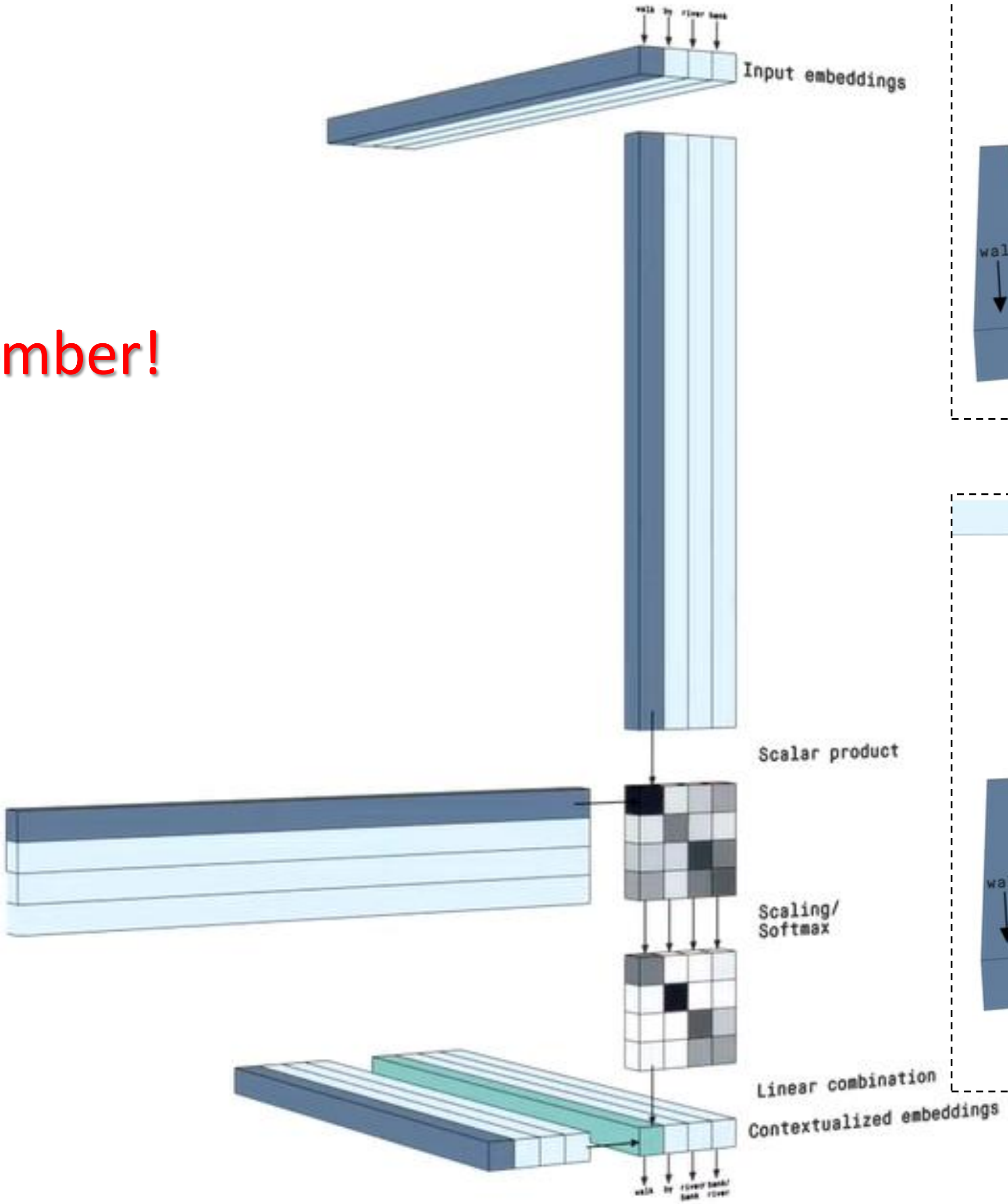
We do the same for I2, I3, etc

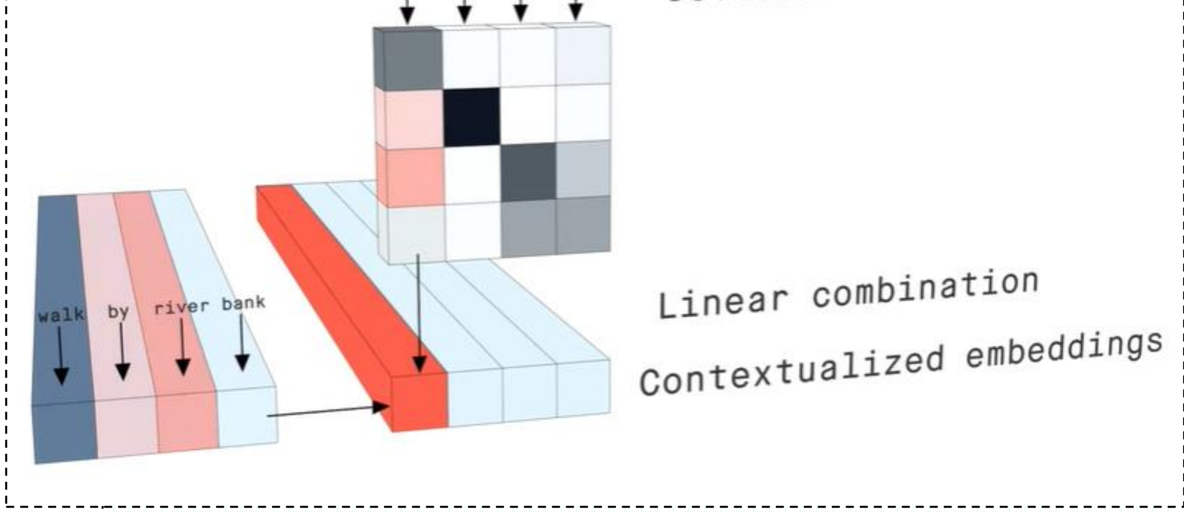
Self Attention

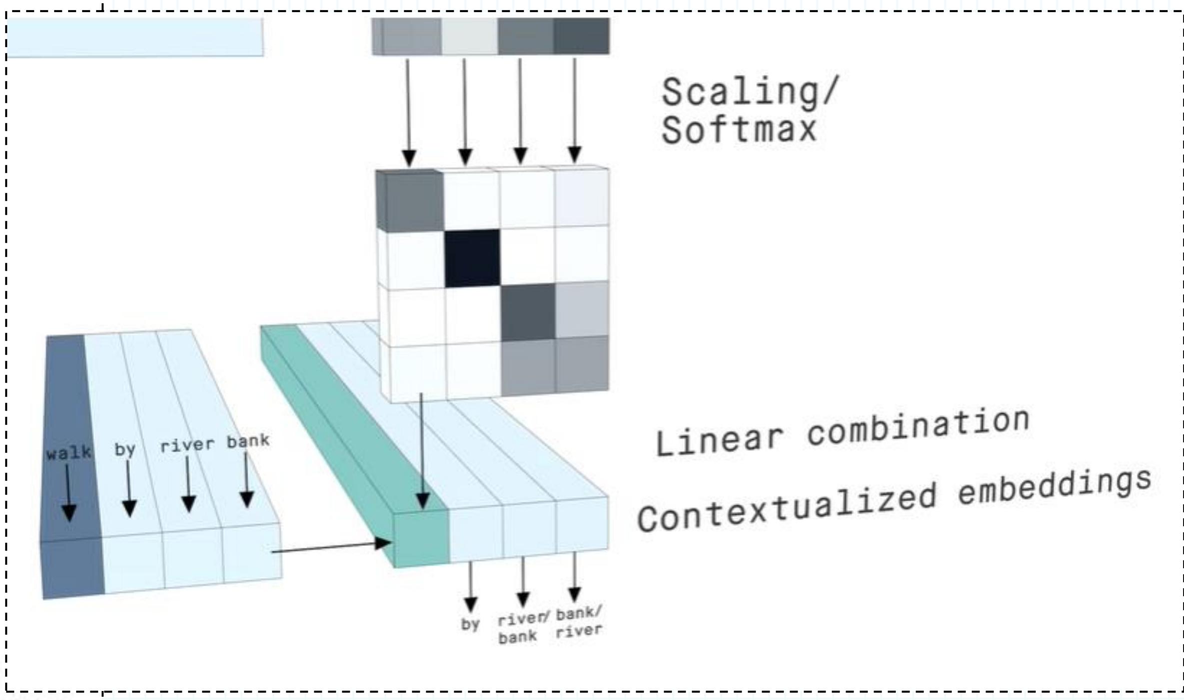
Query Inputs = Key Inputs = Value Inputs



Remember!

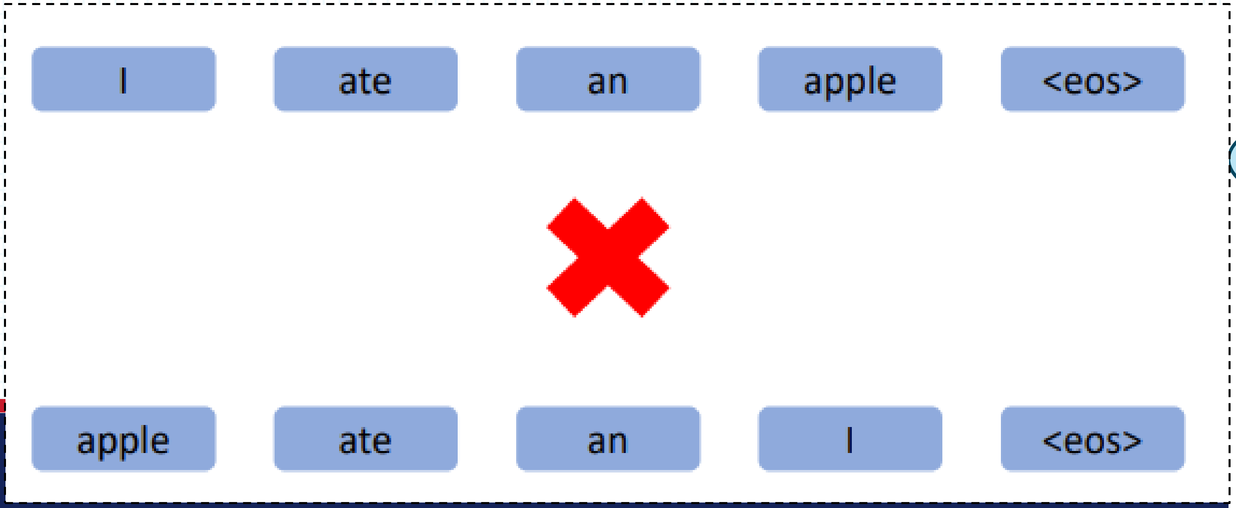

Input embeddings
Scalar product
Scaling/
Softmax
Linear combination
Contextualized embeddings

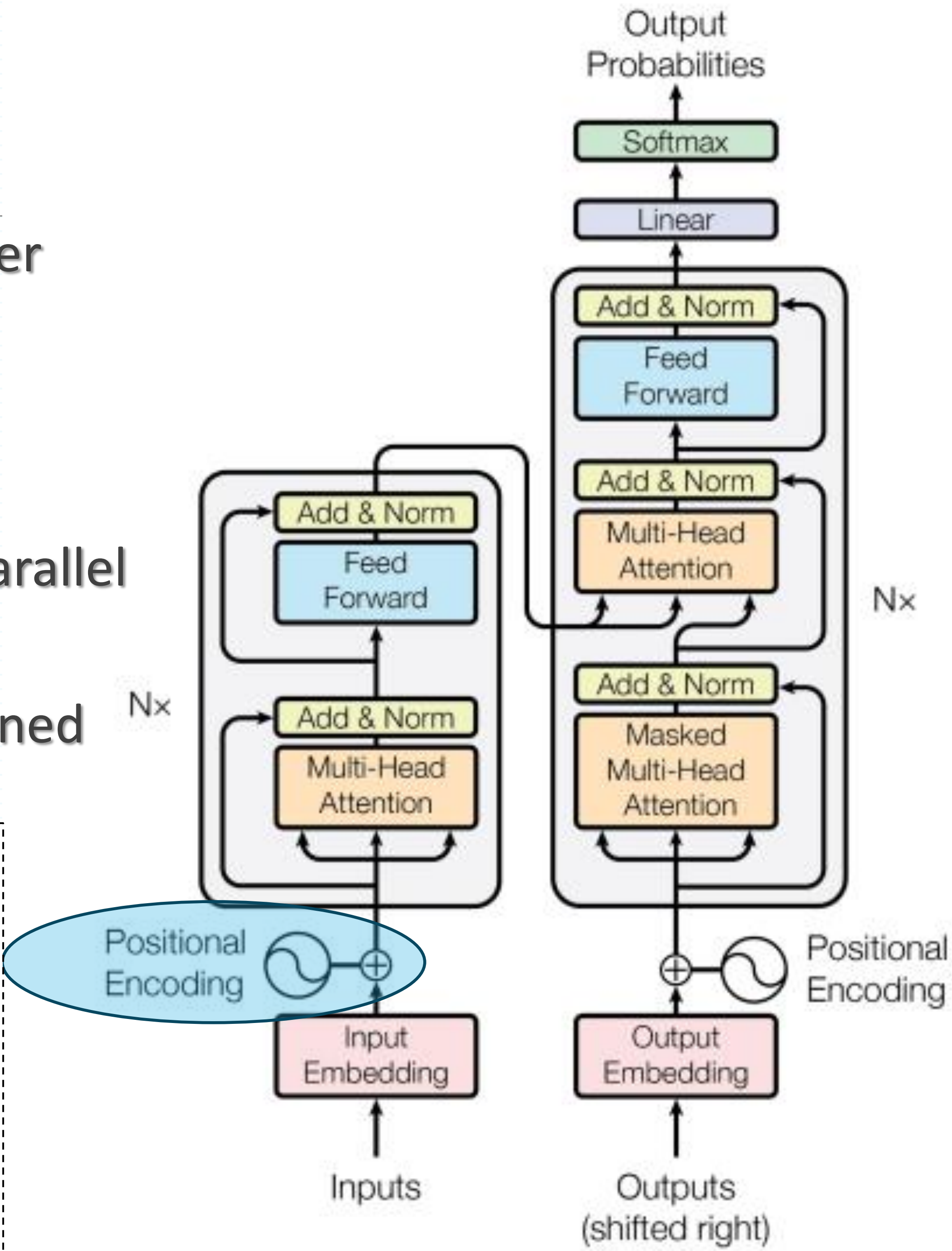

walk
by
river
bank
Linear combination
Contextualized embeddings


Scaling/
Softmax
walk
by
river
bank
Linear combination
Contextualized embeddings
by
river/
bank
bank/
river

Positional Encoding

- Injects sequence order information into transformer models.
- Added to token embeddings to help the model understand word positions.
- Needed because transformers process tokens in parallel and lack inherent order sensitivity.
- Can be fixed (e.g., sinusoidal – sine, cosine) or learned during training.





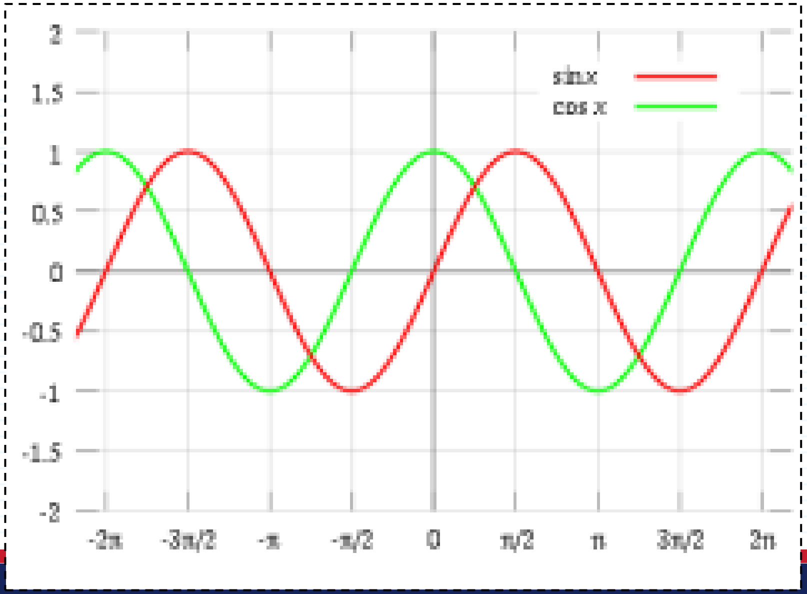
Positional Encoding

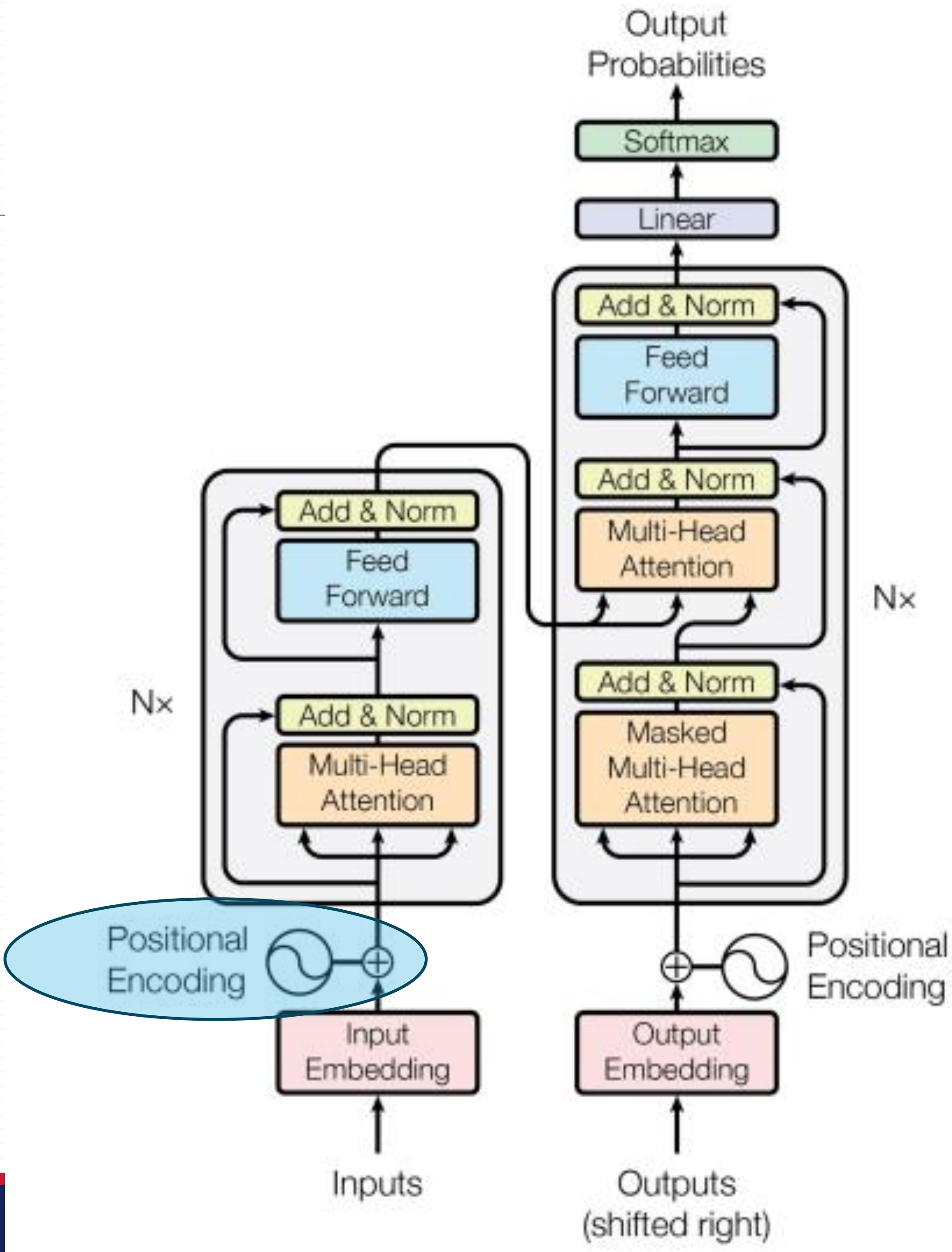
pos -> idx of the token in input sentence

i -> i^{th} dimension out of d

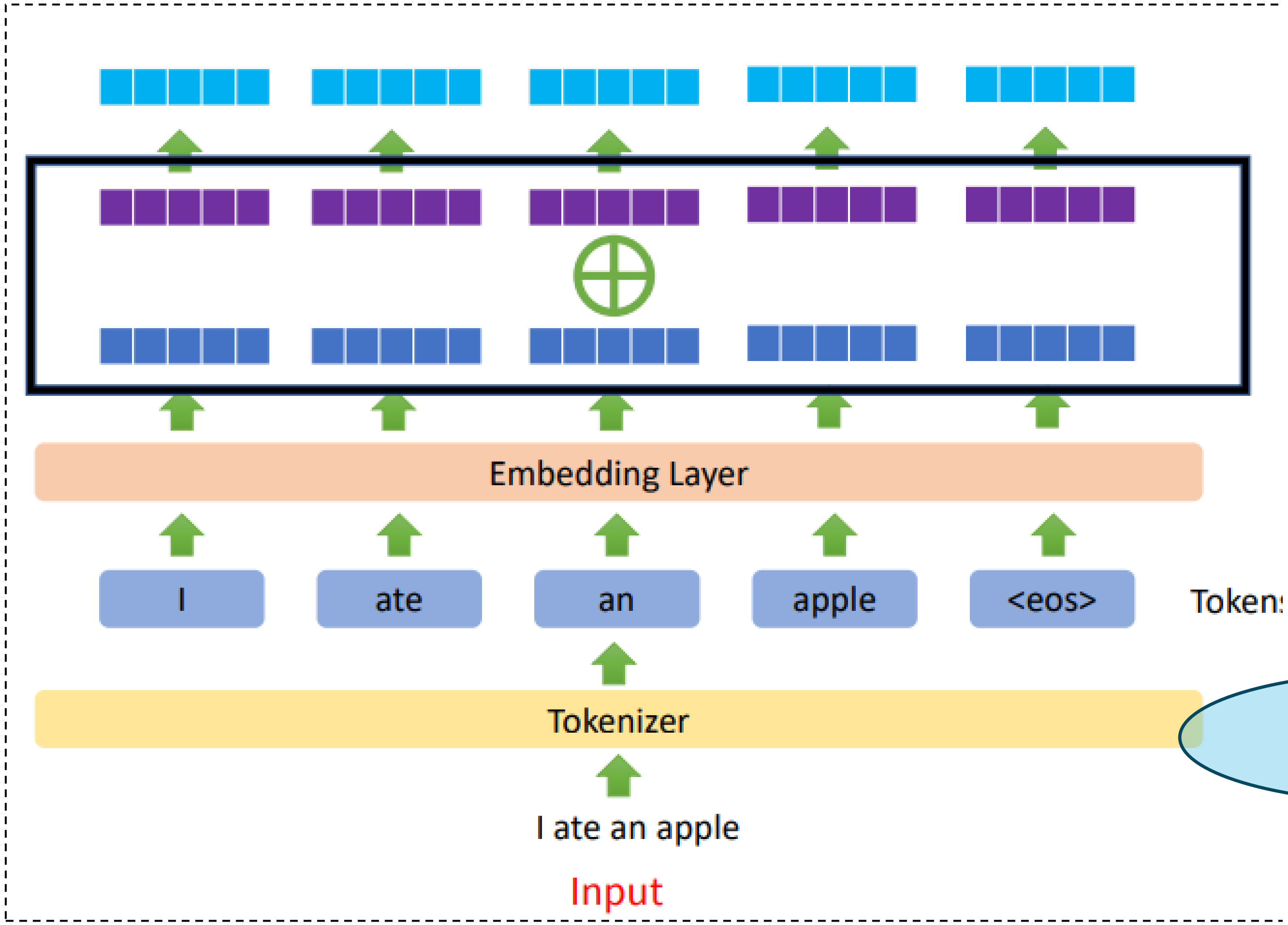
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

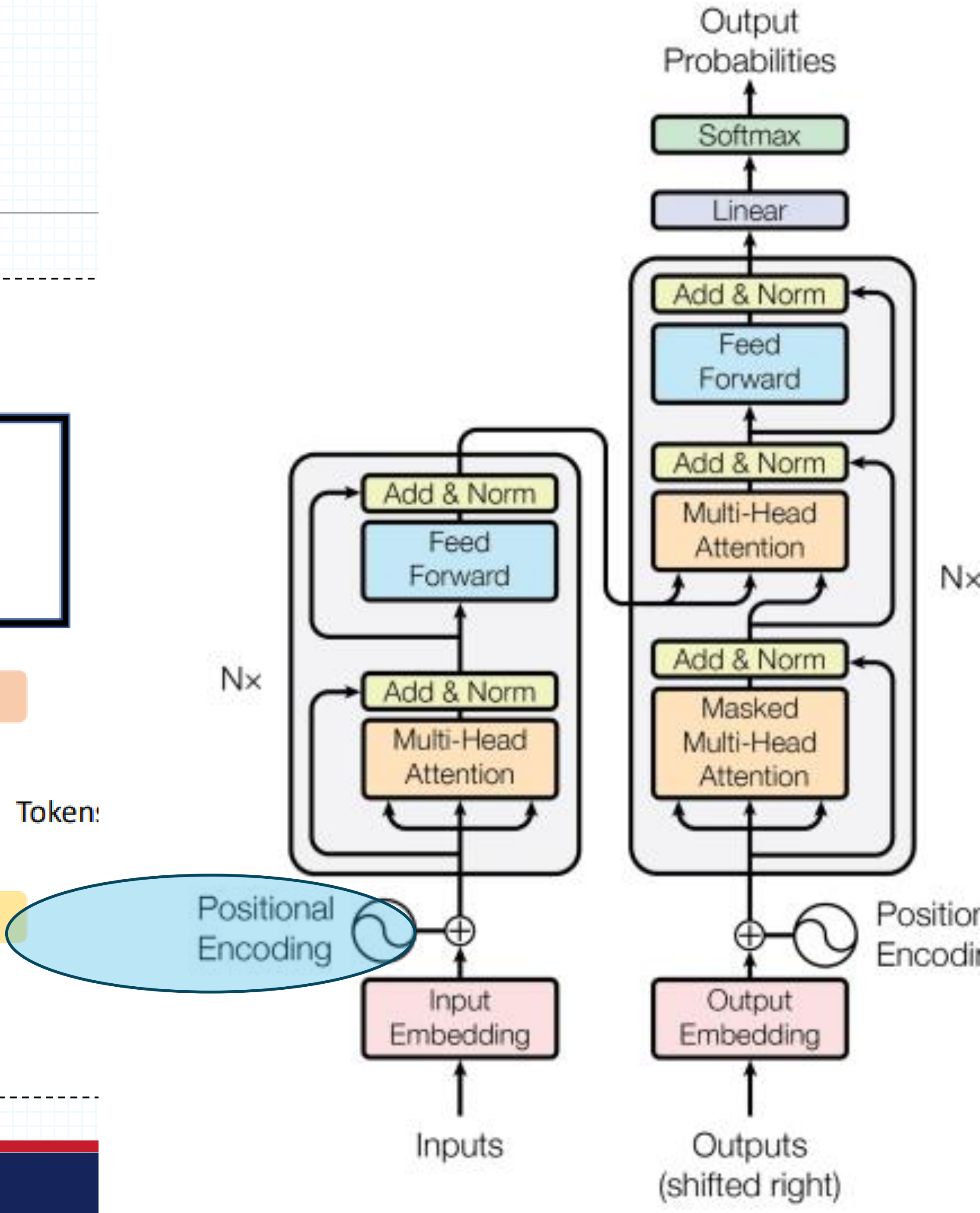
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$



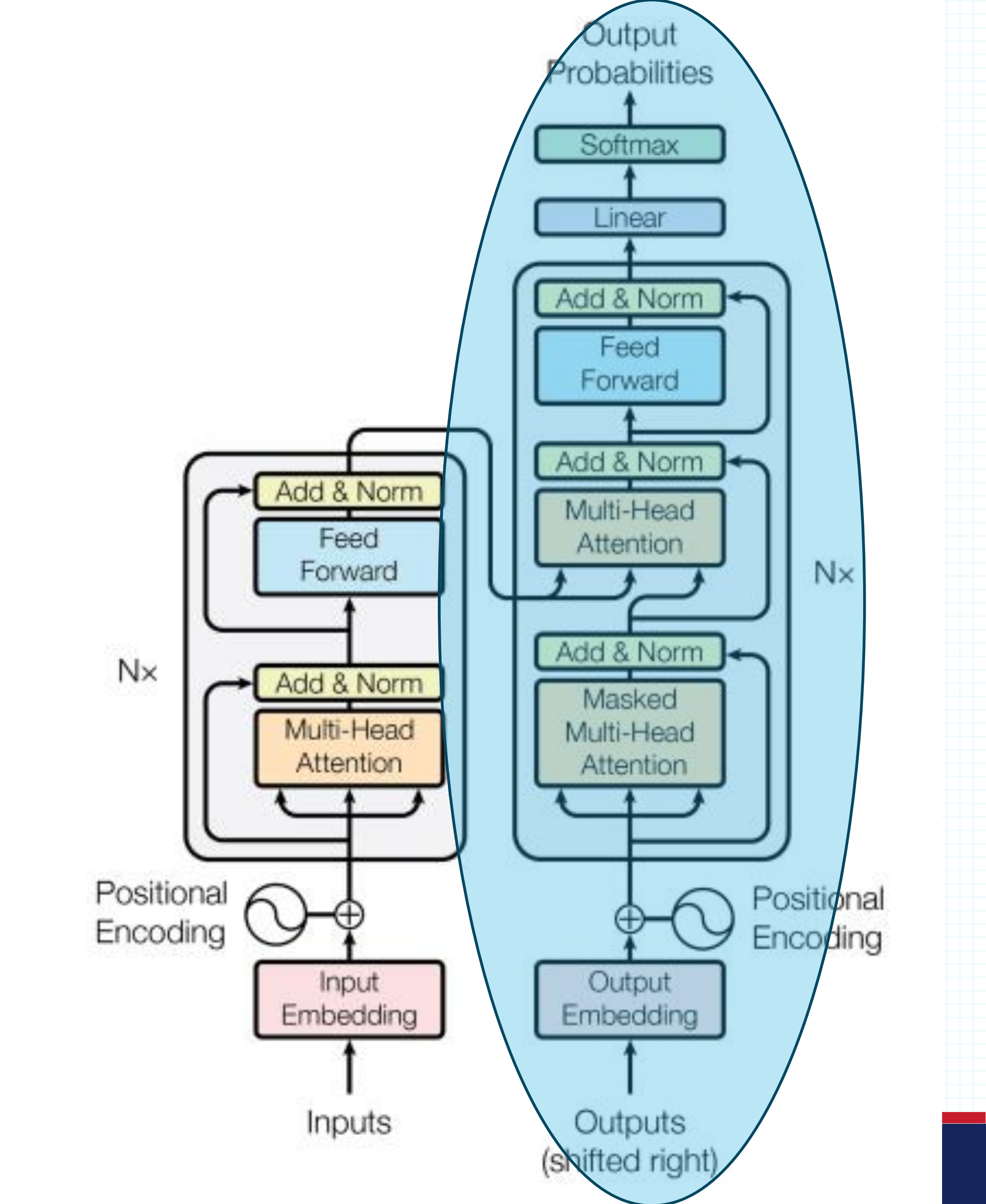


Position Embedding



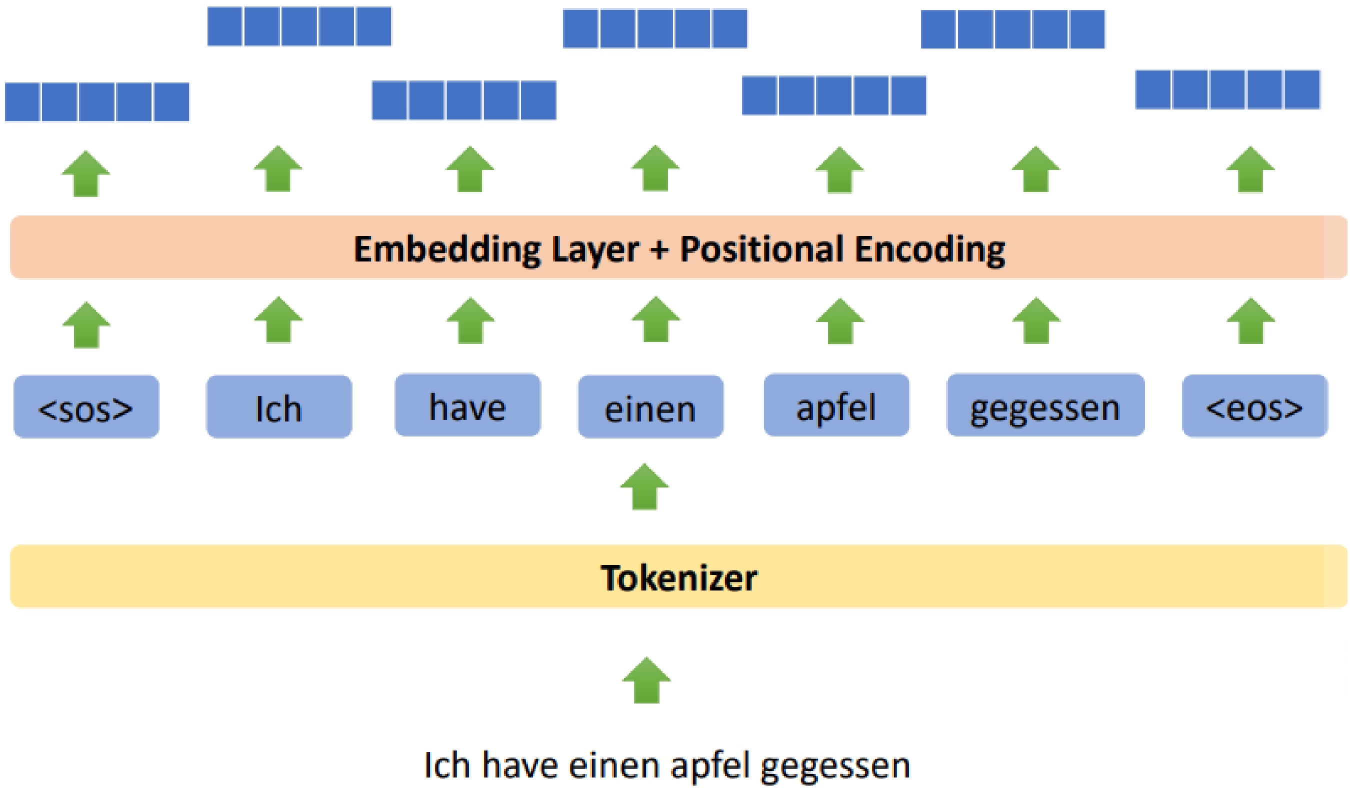


Next..

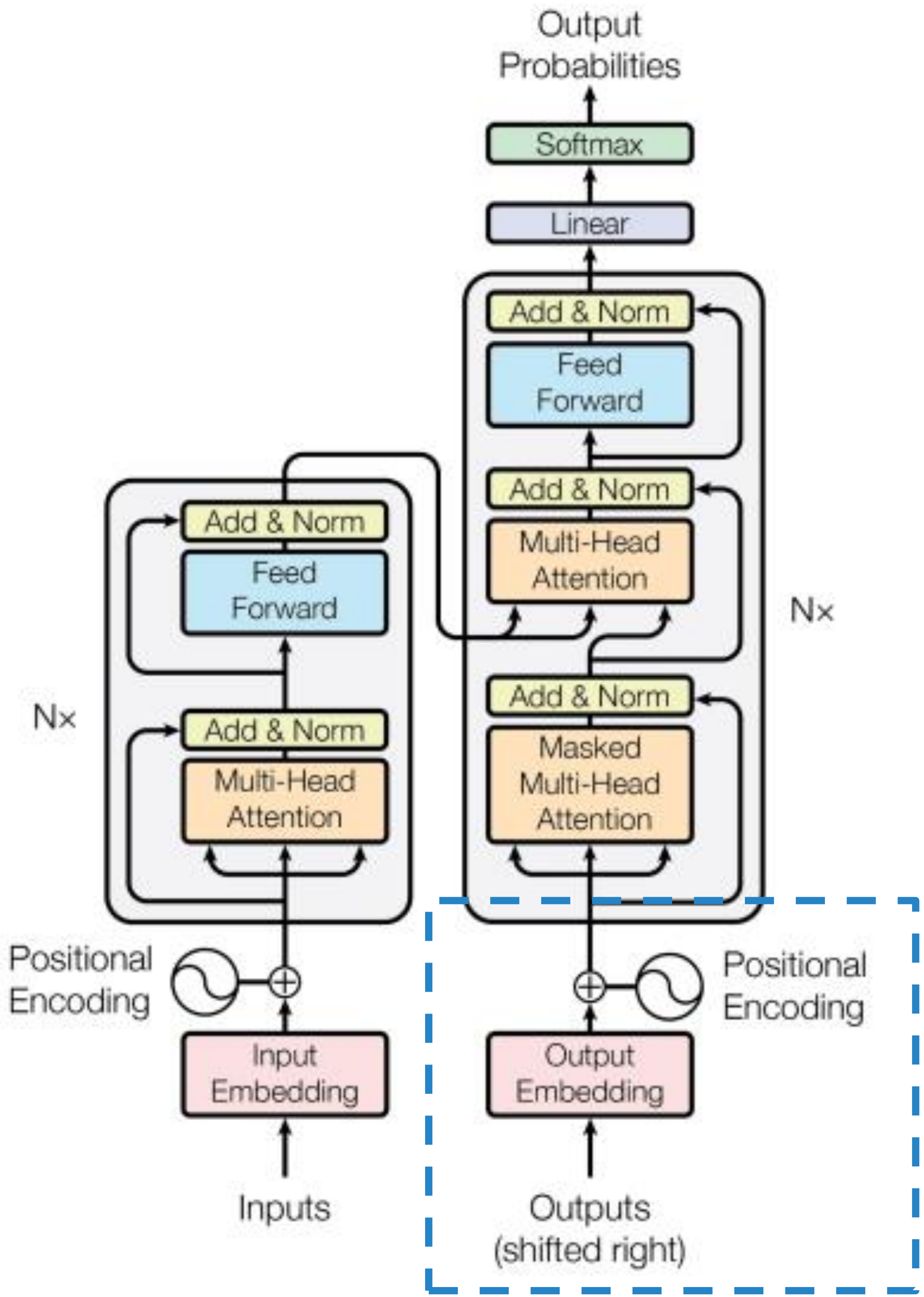


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

Output Embedding



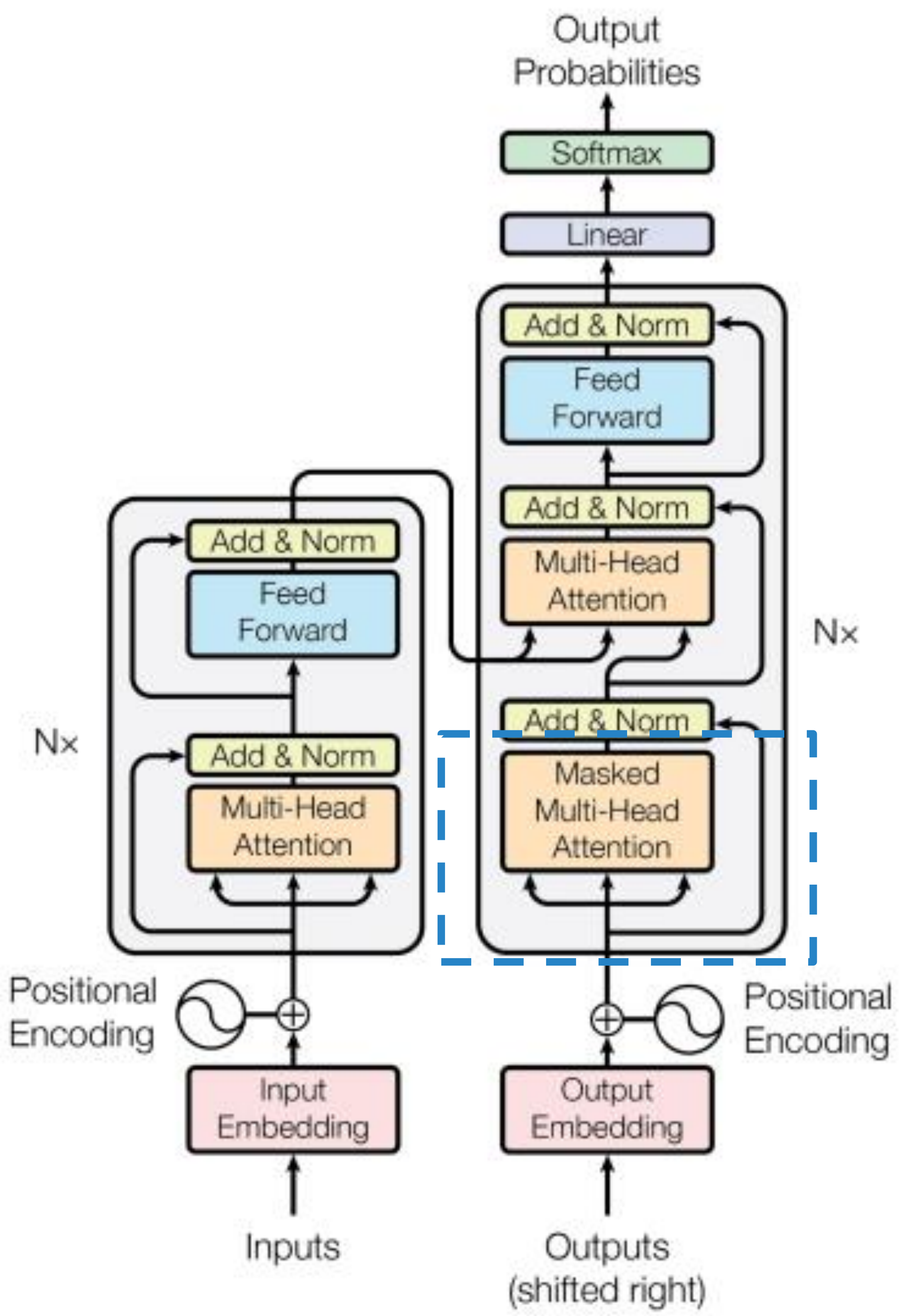
Generate Target Emebeddings



Masked Attention

1	<sos>	Ich	have	einen	apfel	gegessen	<eos>
2	<sos>	Ich	have	einen	apfel	gegessen	<eos>
3	<sos>	Ich	have	einen	apfel	gegessen	<eos>
4	<sos>	Ich	have	einen	apfel	gegessen	<eos>
5	<sos>	Ich	have	einen	apfel	gegessen	<eos>
6	<sos>	Ich	have	einen	apfel	gegessen	<eos>
7	<sos>	Ich	have	einen	apfel	gegessen	<eos>

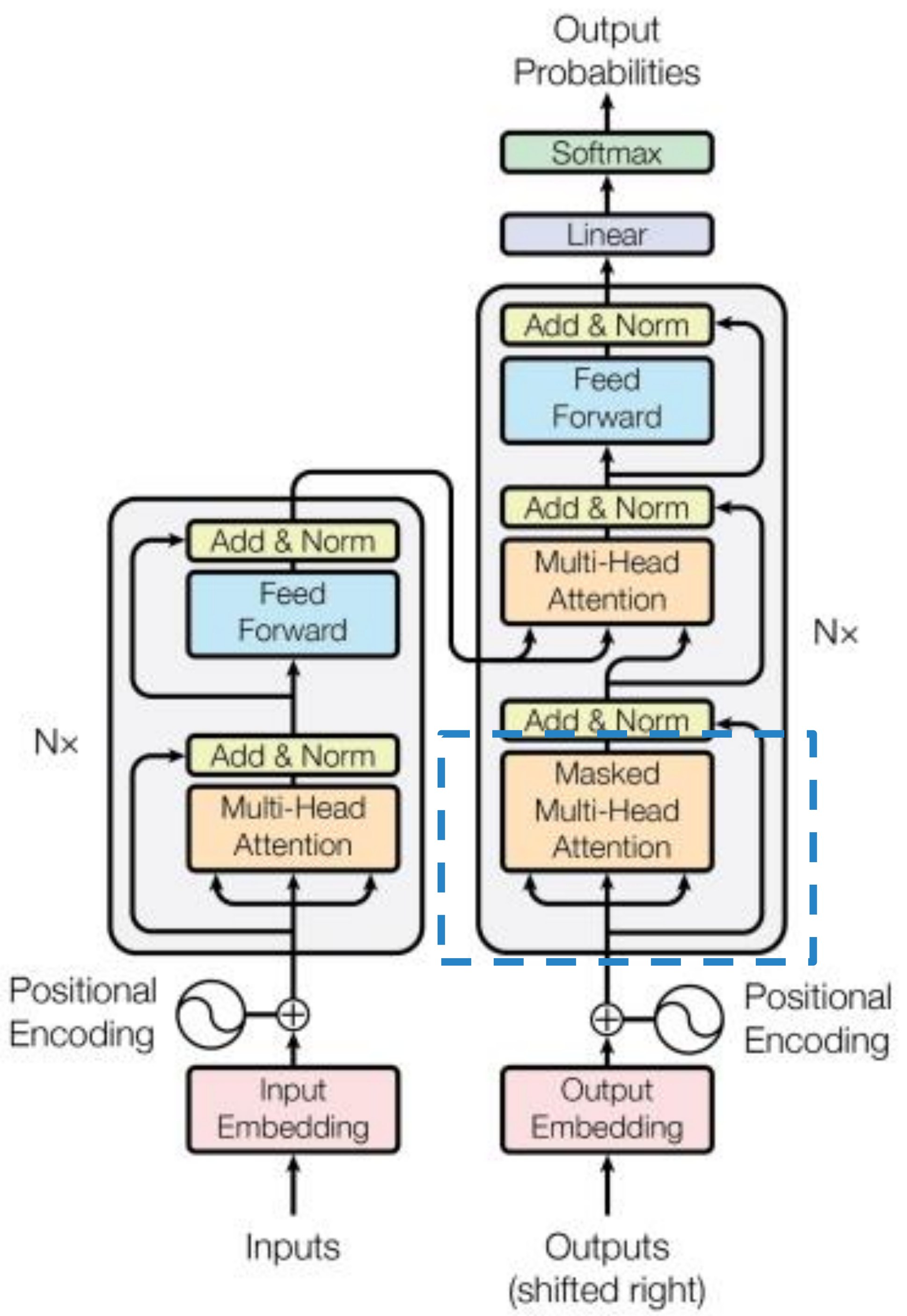
Mask the available attention values ?



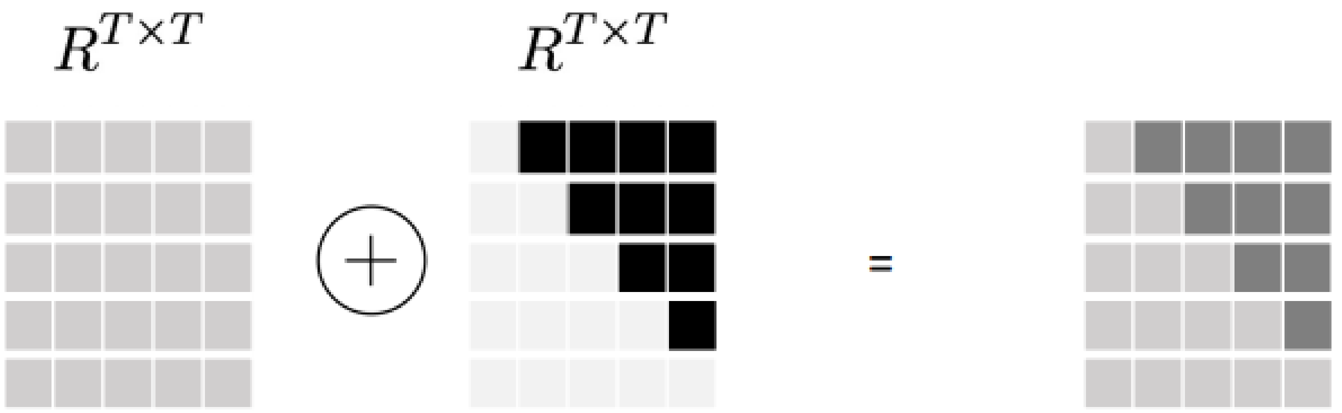
Masked Attention

1	<sos>	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
2	<sos>	Ich	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$
3	<sos>	Ich	have	$-\infty$	$-\infty$	$-\infty$	$-\infty$
4	<sos>	Ich	have	einen	$-\infty$	$-\infty$	$-\infty$
5	<sos>	Ich	have	einen	apfel	$-\infty$	$-\infty$
6	<sos>	Ich	have	einen	apfel	gegessen	$-\infty$
7	<sos>	Ich	have	einen	apfel	gegessen	<eos>

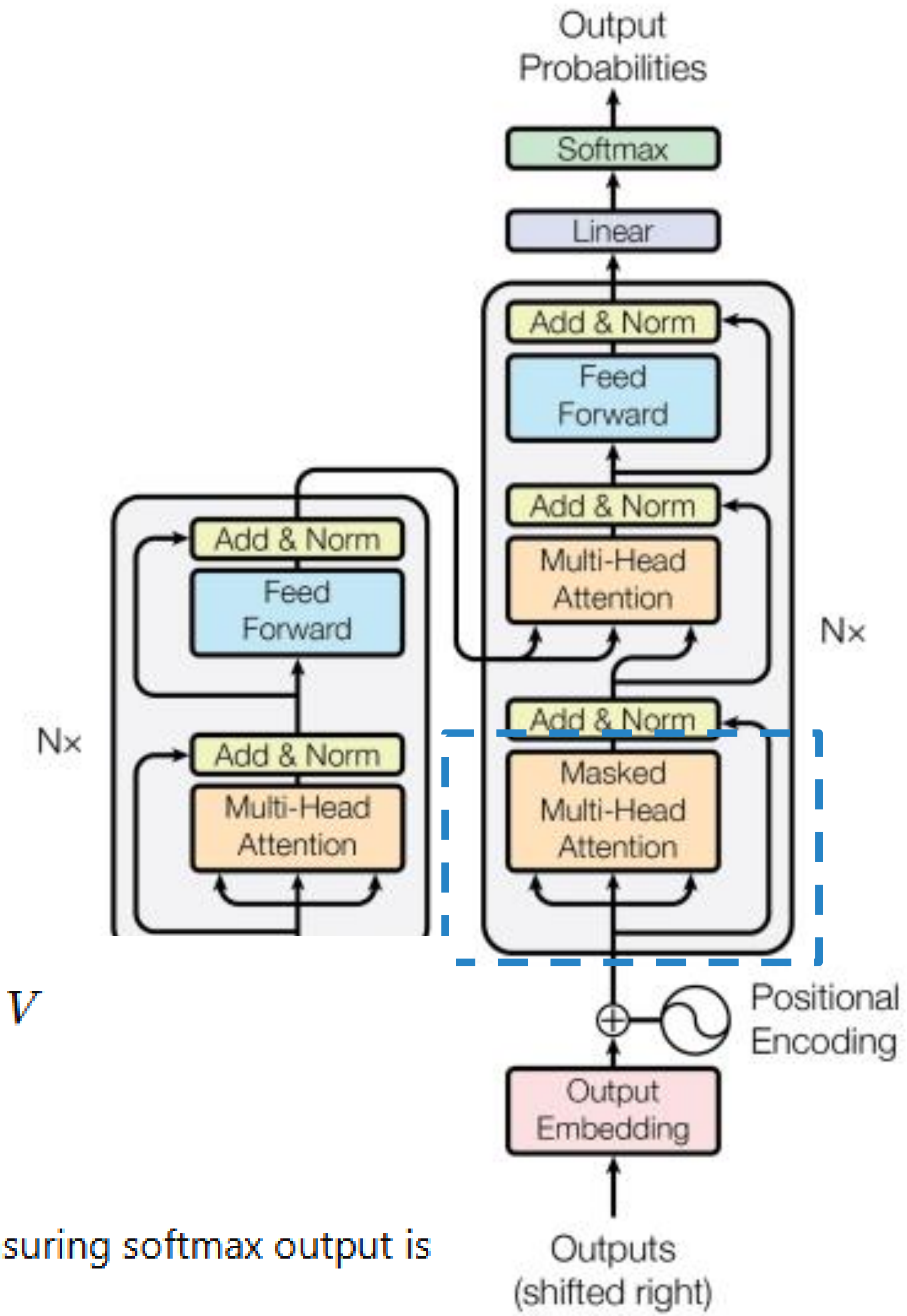
Softmax -> $-\infty$ -> 0



Masked Attention



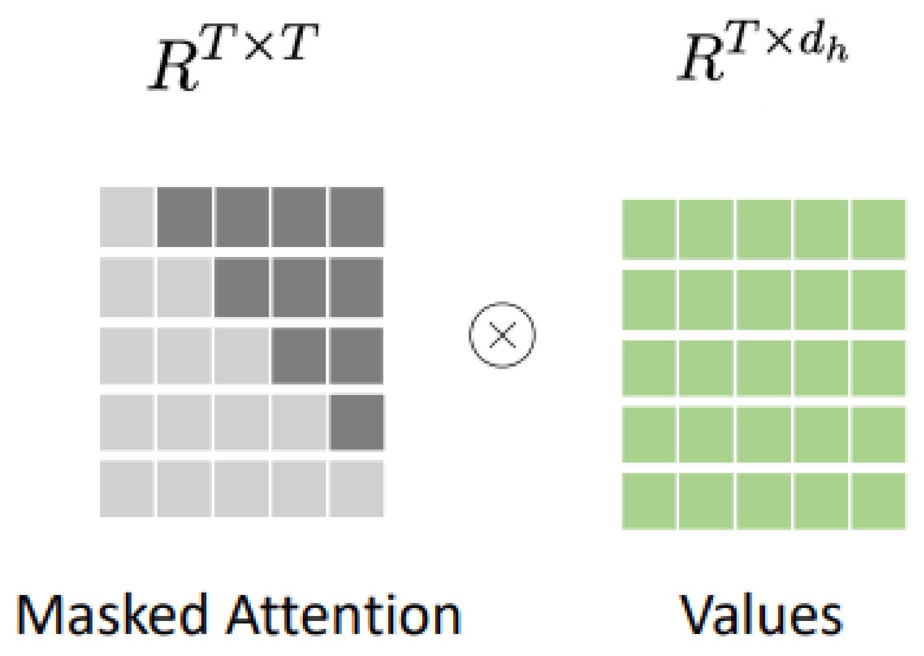
QK^T Attention Mask: M Masked Attention

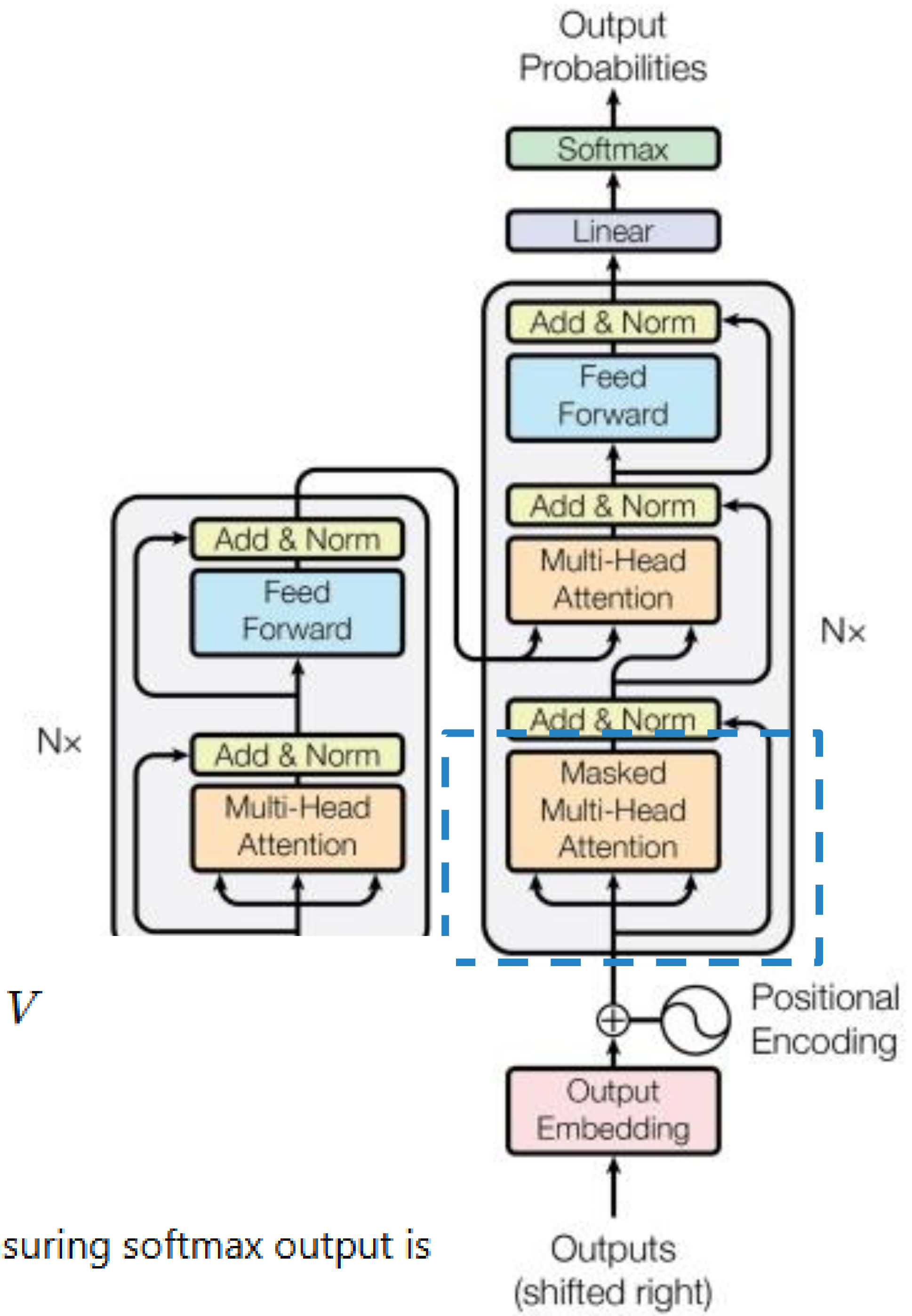


$$\text{Attention}(Q,K,V)=\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}+M\right)V$$

- $Q,K,V\in\mathbb{R}^{T\times d_k}$: represent queries, keys, and values.
- M: a mask matrix with $-\infty$ in positions corresponding to future tokens, ensuring softmax output is zero for those.

Masked Attention

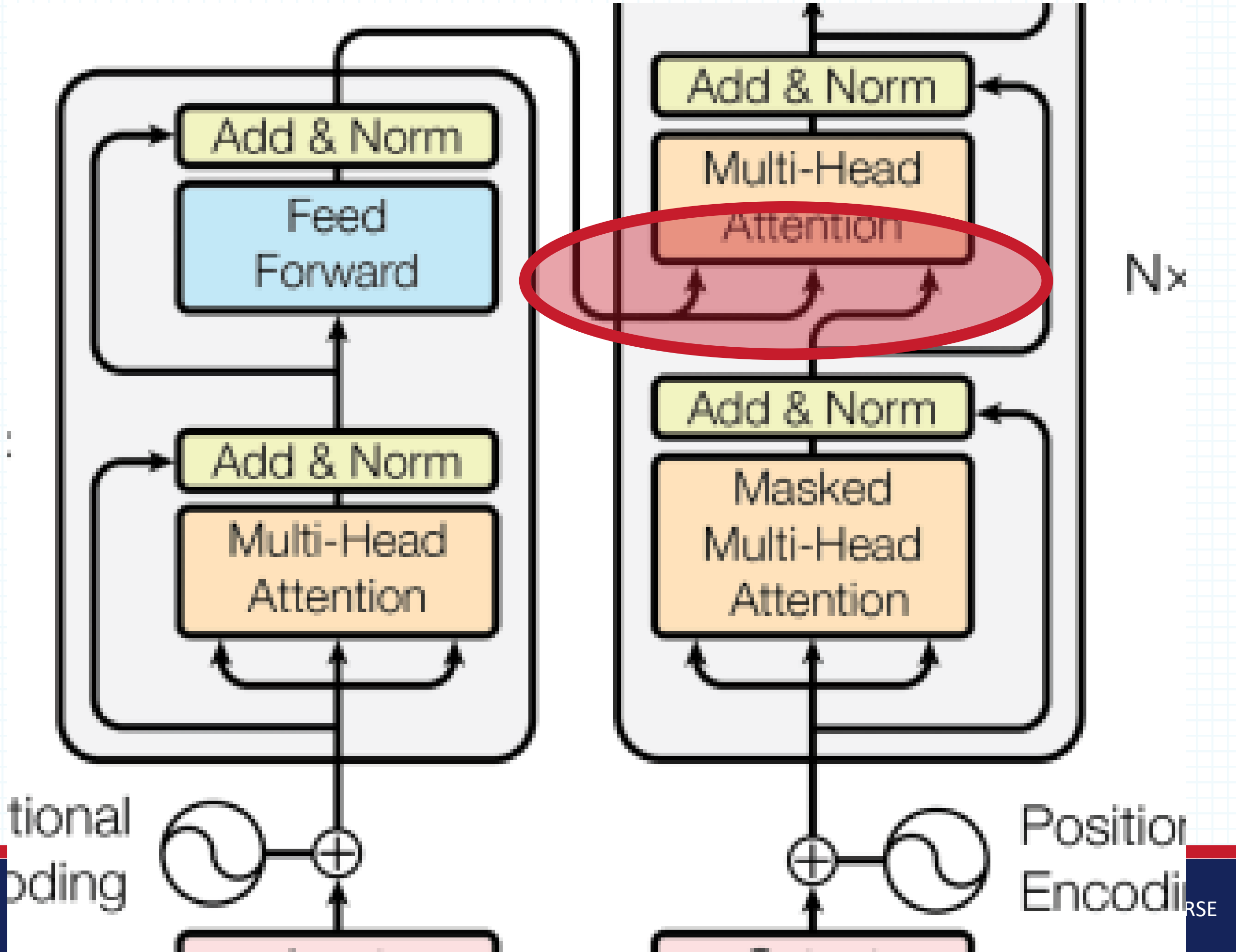


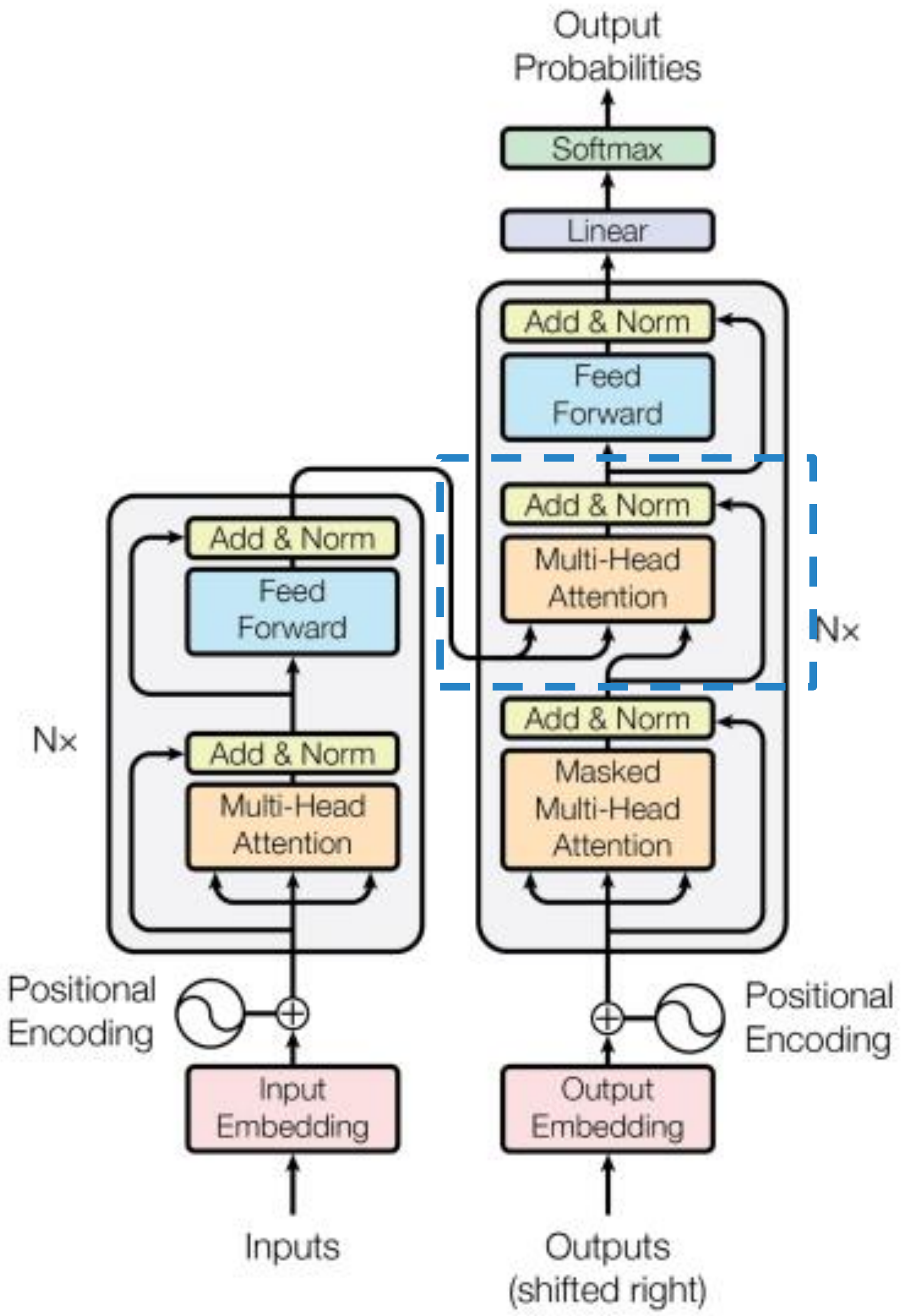


$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V$$

- $Q, K, V \in \mathbb{R}^{T\times d_k}$: represent queries, keys, and values.
- M: a mask matrix with $-\infty$ in positions corresponding to future tokens, ensuring softmax output is zero for those.

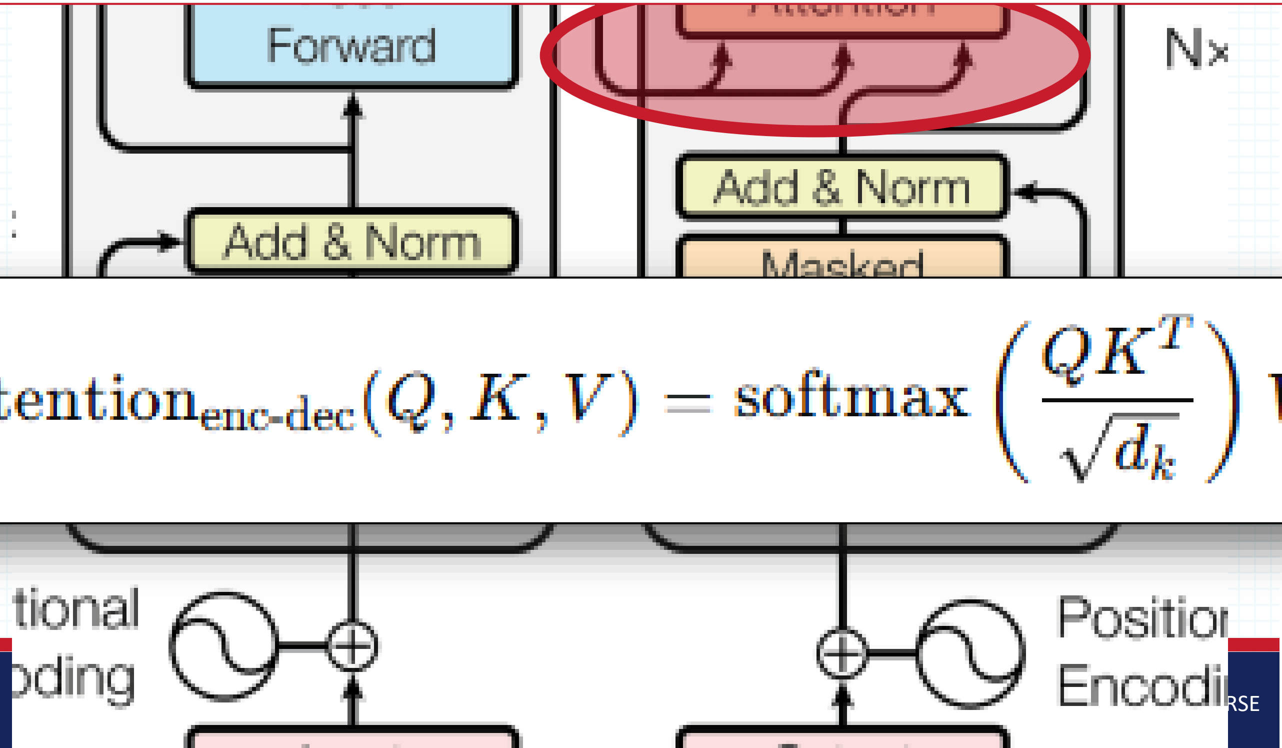
Encoder-Decoder Attention (Cross-Attention)



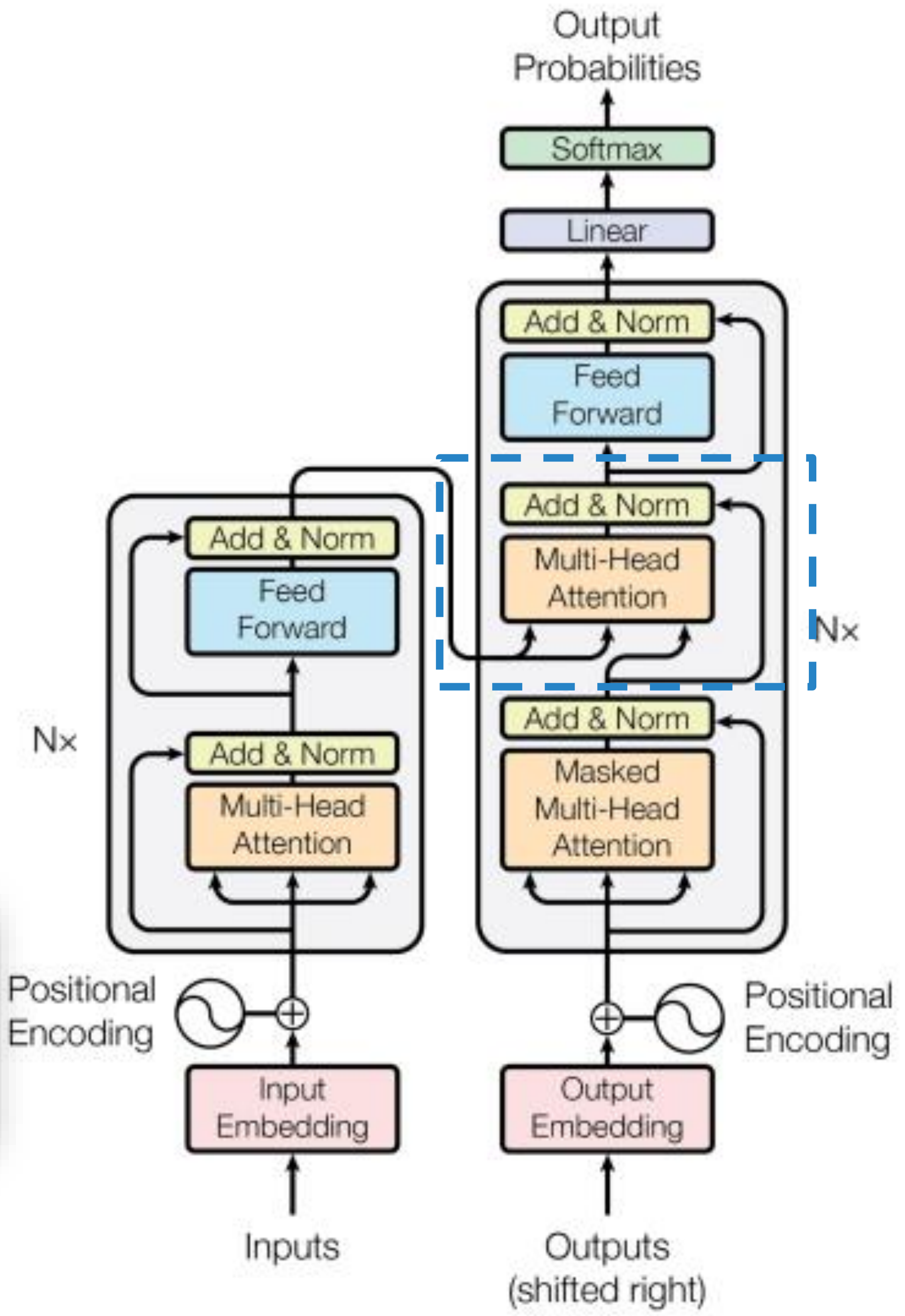


Encoder-Decoder Attention (Cross-Attention)

The outputs of the **Encoder** act as Keys and Values, and the output from the **Decoder's** self-attention acts as Queries.



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

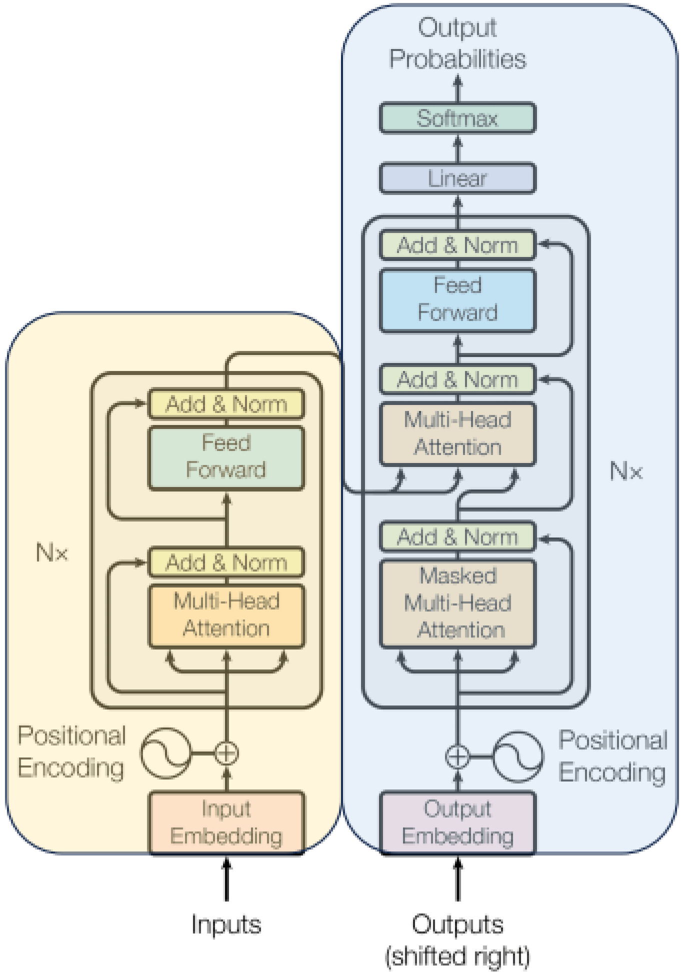


Transformers

- a Transformer architecture can be designed in **three different ways** depending on the task and objective:
- 1. **Encoder-Only Transformers**
- **2. Decoder-Only Transformers**
- 3. **Encoder-Decoder Transformers (Seq2Seq)**

BERT
Oct 2018

Representation



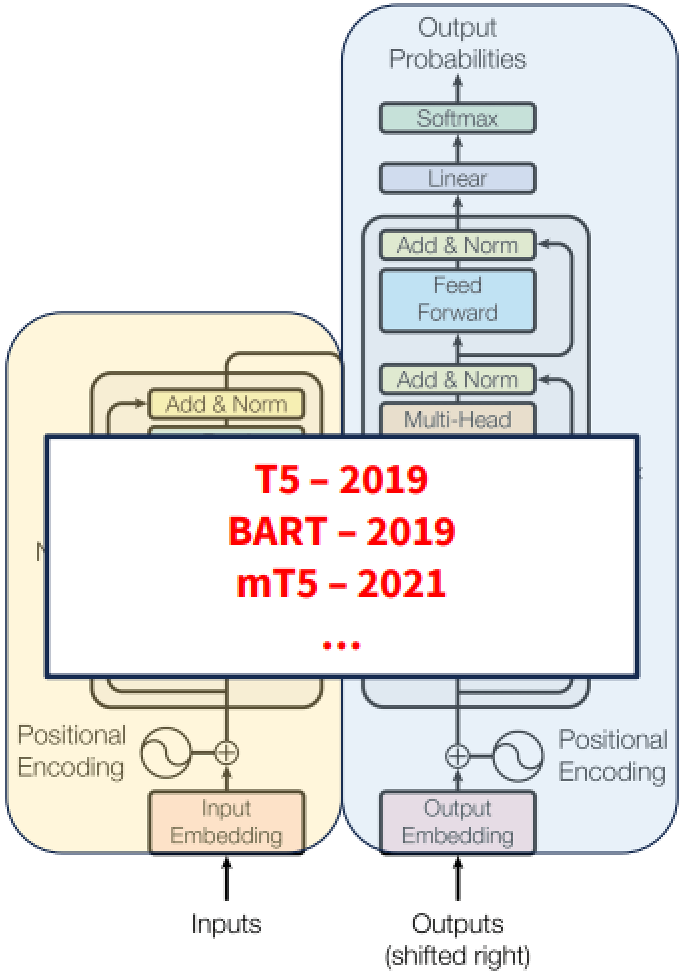
GPT
Jun 2018

Generation

Transformers… The LLM Era

BERT – 2018
DistilBERT – 2019
RoBERTa – 2019
ALBERT – 2019
ELECTRA – 2020
DeBERTa – 2020
…

Representation



GPT – 2018
GPT-2 – 2019
GPT-3 – 2020
GPT-Neo – 2021
GPT-3.5 (ChatGPT) – 2022
LLaMA – 2023
GPT-4 – 2023
…

Generation