



Week 11

السنة الخامسة – هندسة المعلوماتية / الذكاء الصناعي

مقرر التعلم التلقائي

Transformers

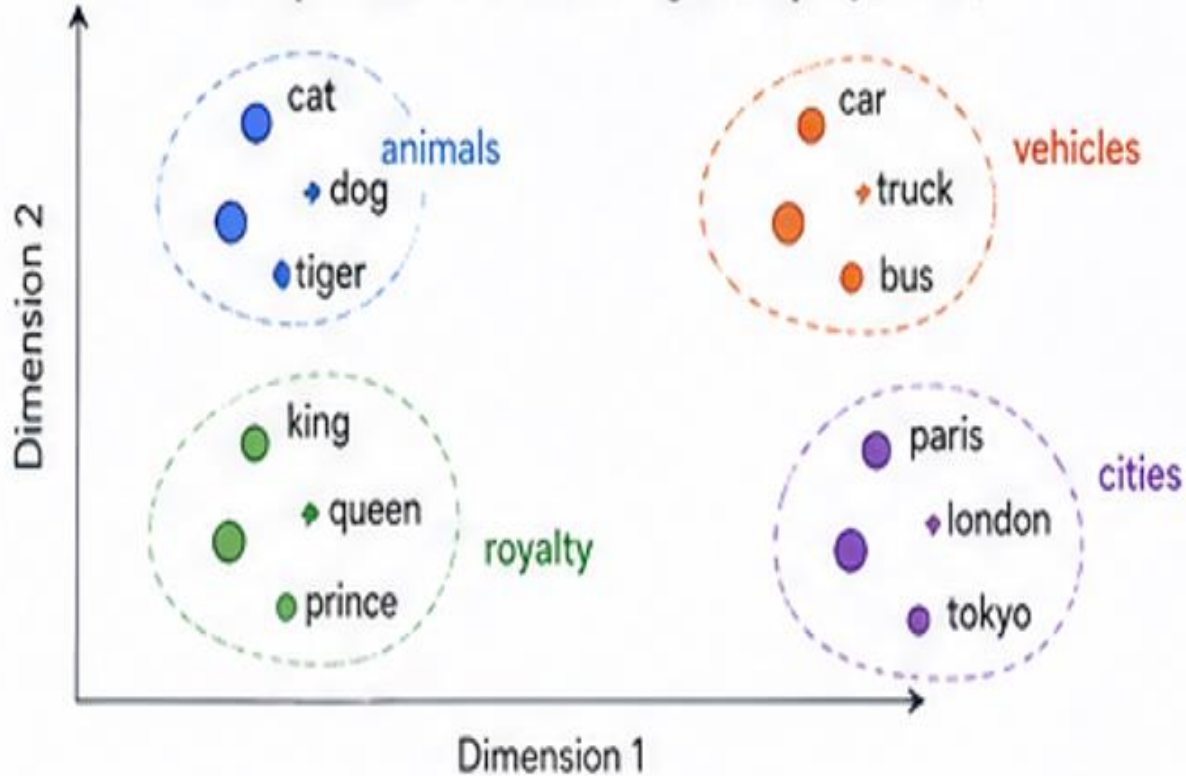
د. رياض سنبل

[Access Course Materials](#) 

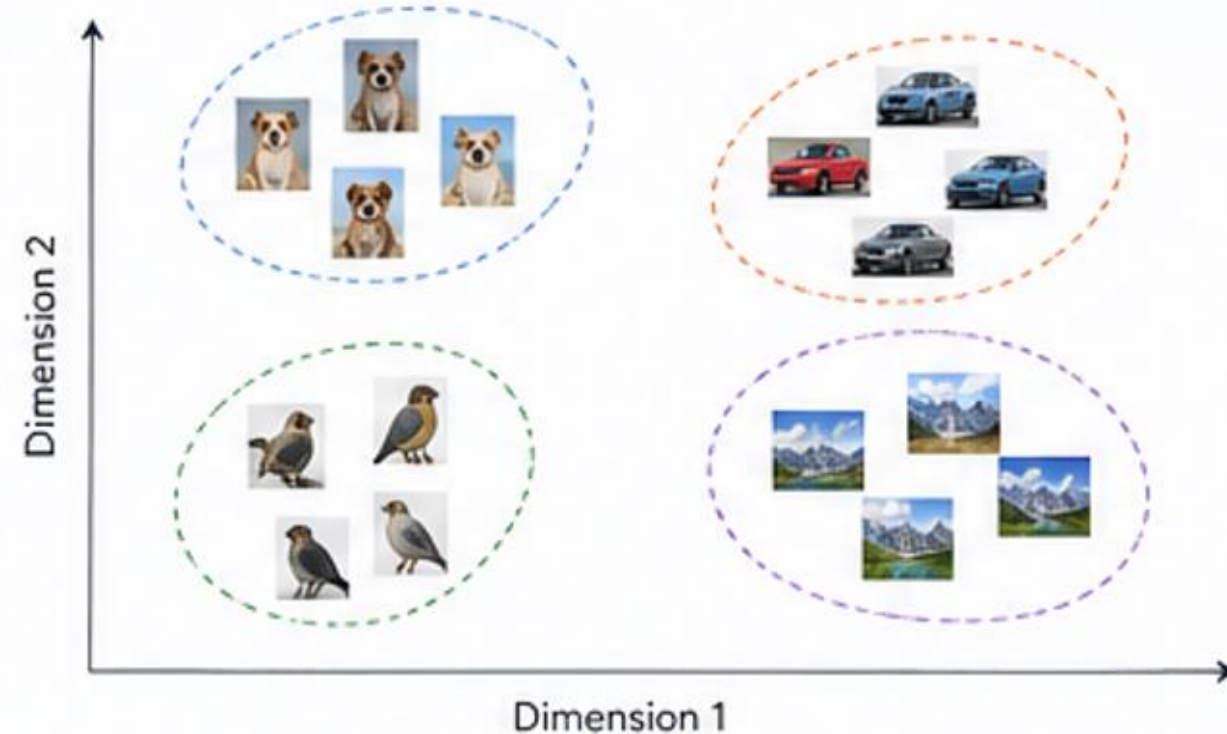
Embeddings: The Foundation of Modern AI

- Embeddings are numerical representations that capture meaning.

Example: Word Embeddings (2D projection)



Example: Image Embeddings (2D projection)



- Words with similar meanings are represented by nearby vectors.

Embeddings are used for many types of data

Text Embeddings



Used in:

- Chatbots
- Search
- Translation
- LLMs



"I love reading books."

→ vector

Image Embeddings



Used in:

- Image search
- Classification
- Recommendations
- Retrieval



(image)

→ vector

Audio Embeddings



Used in:

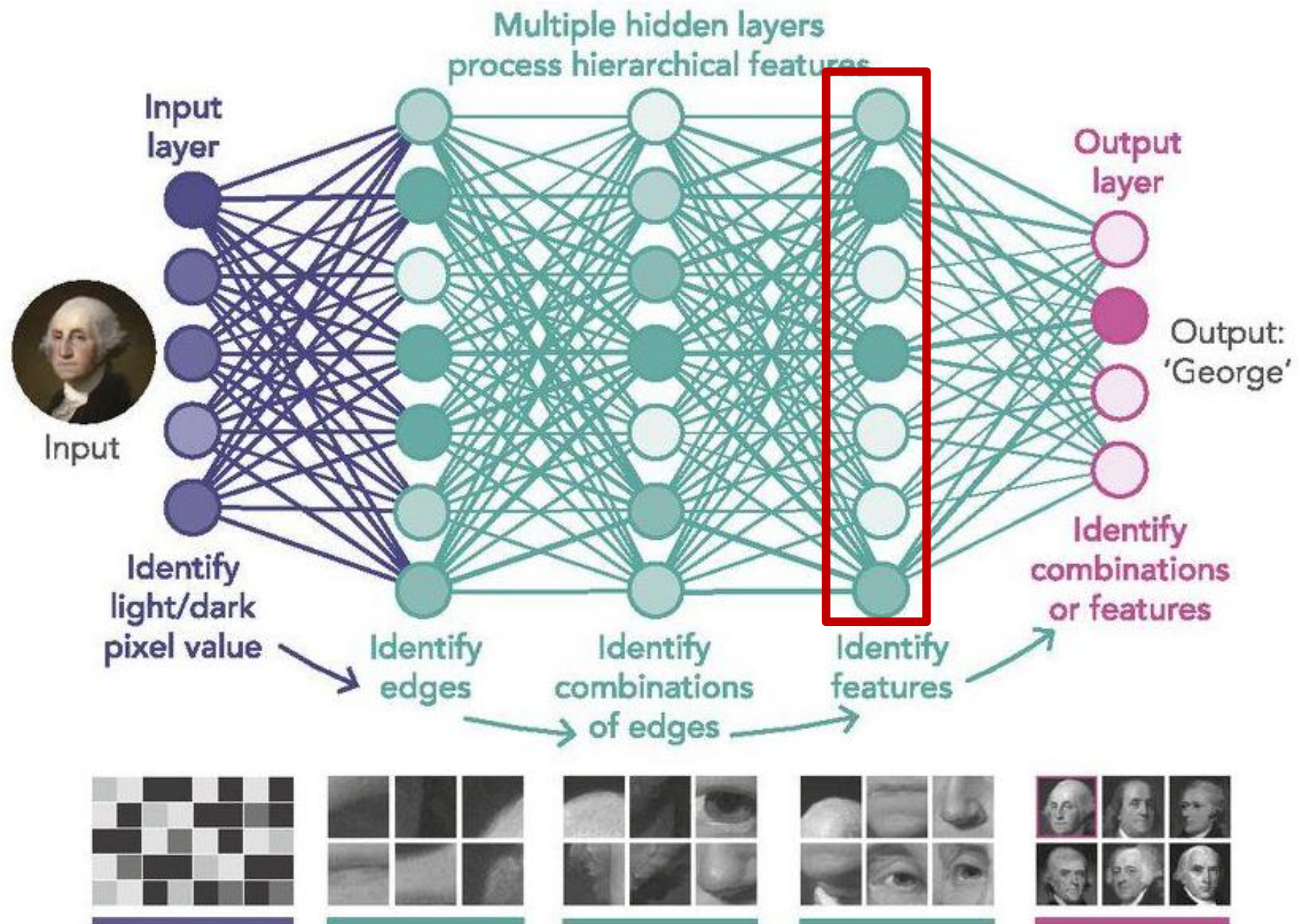
- Speech recognition
- Speaker ID
- Music recommendation
- Sound understanding



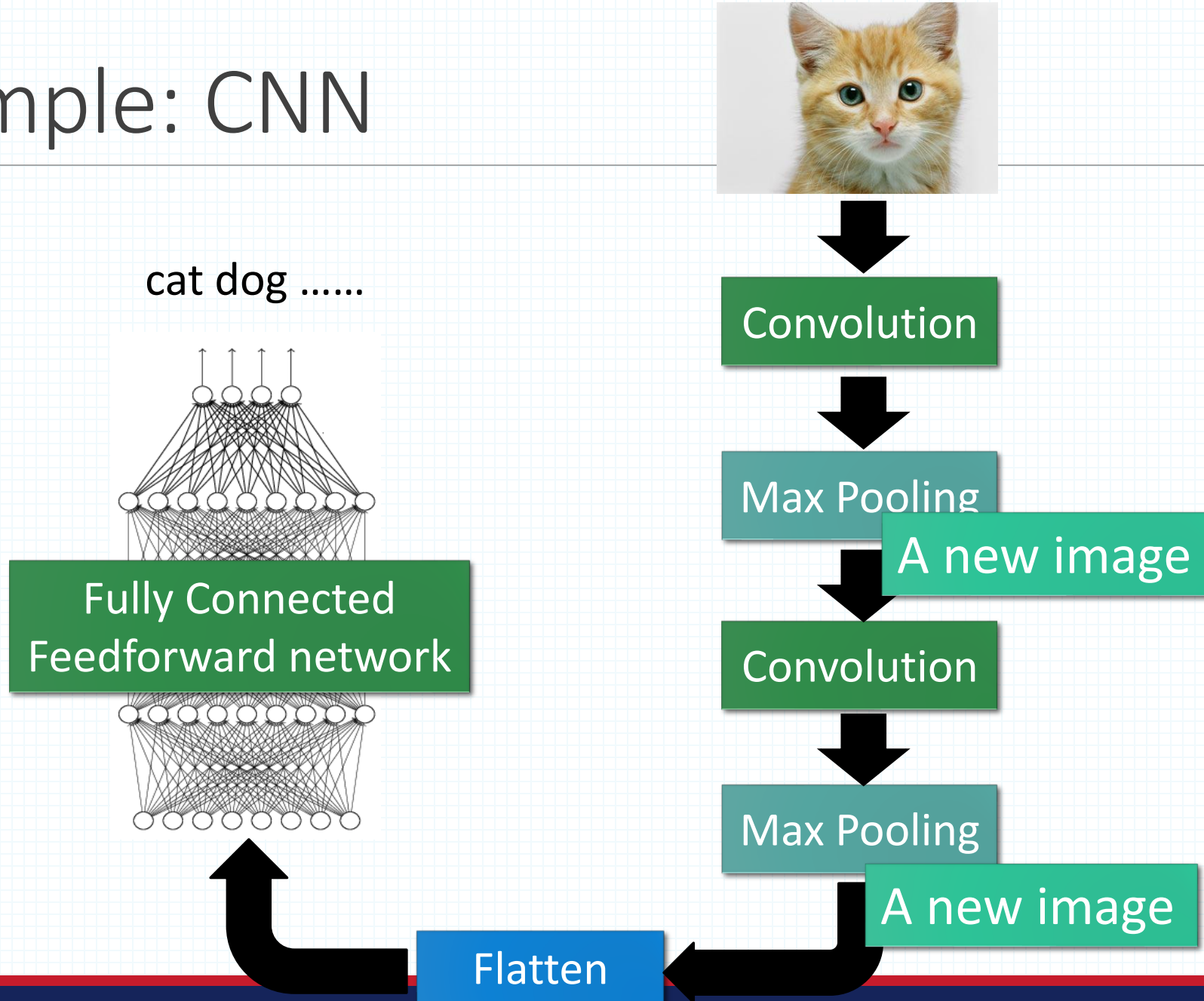
(audio)

→ vector

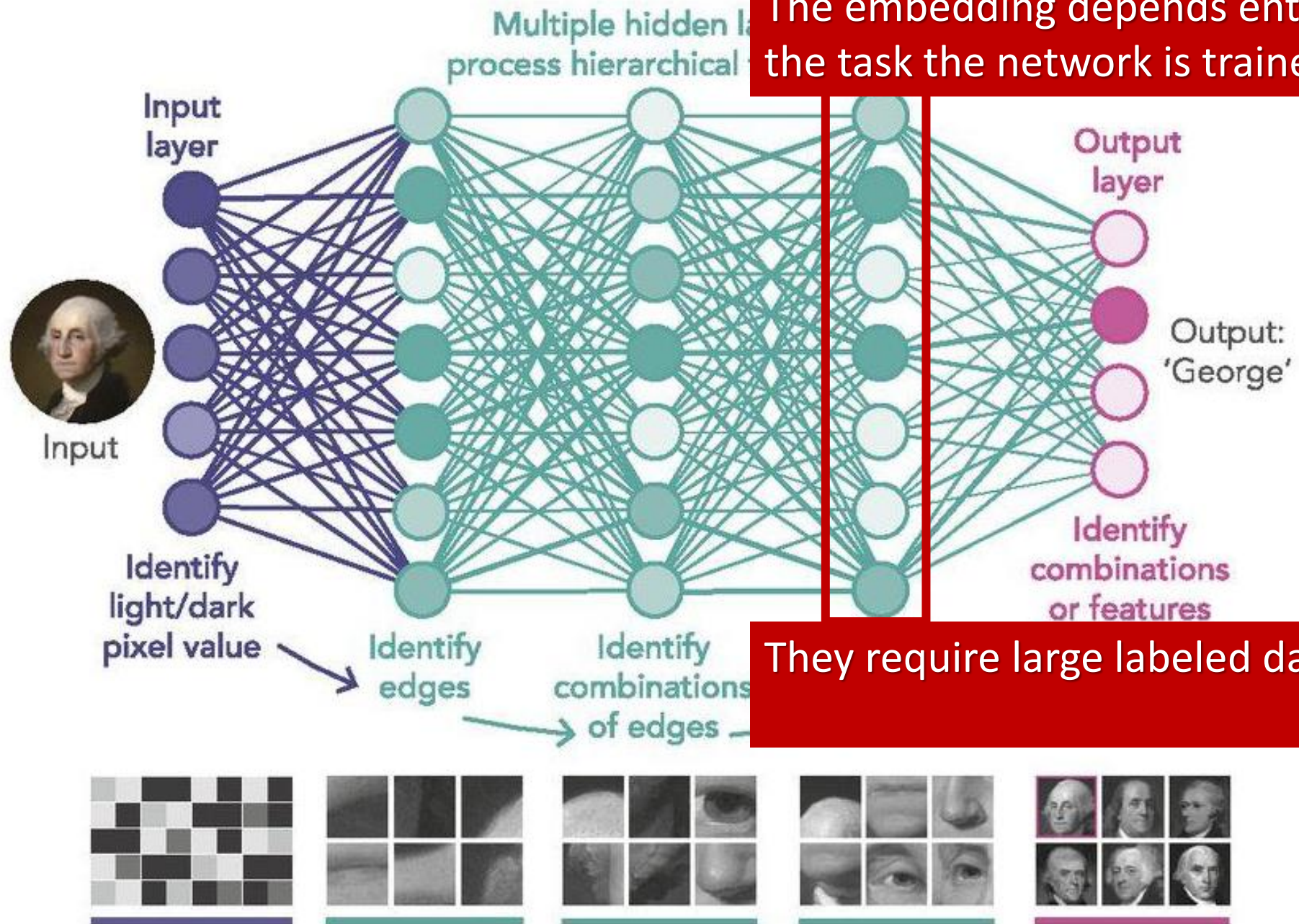
How to build these Embedding Vectors



Example: CNN



BUT...



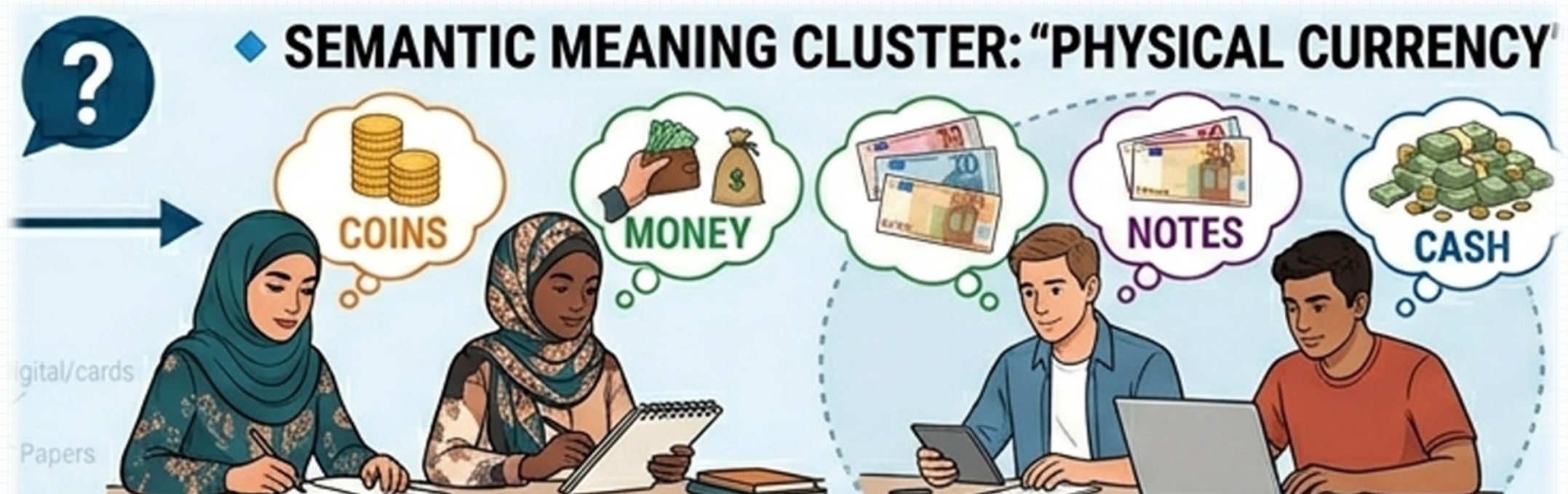
The embedding depends entirely on the task the network is trained on.

They require large labeled datasets!

Solution: Self-supervised learning

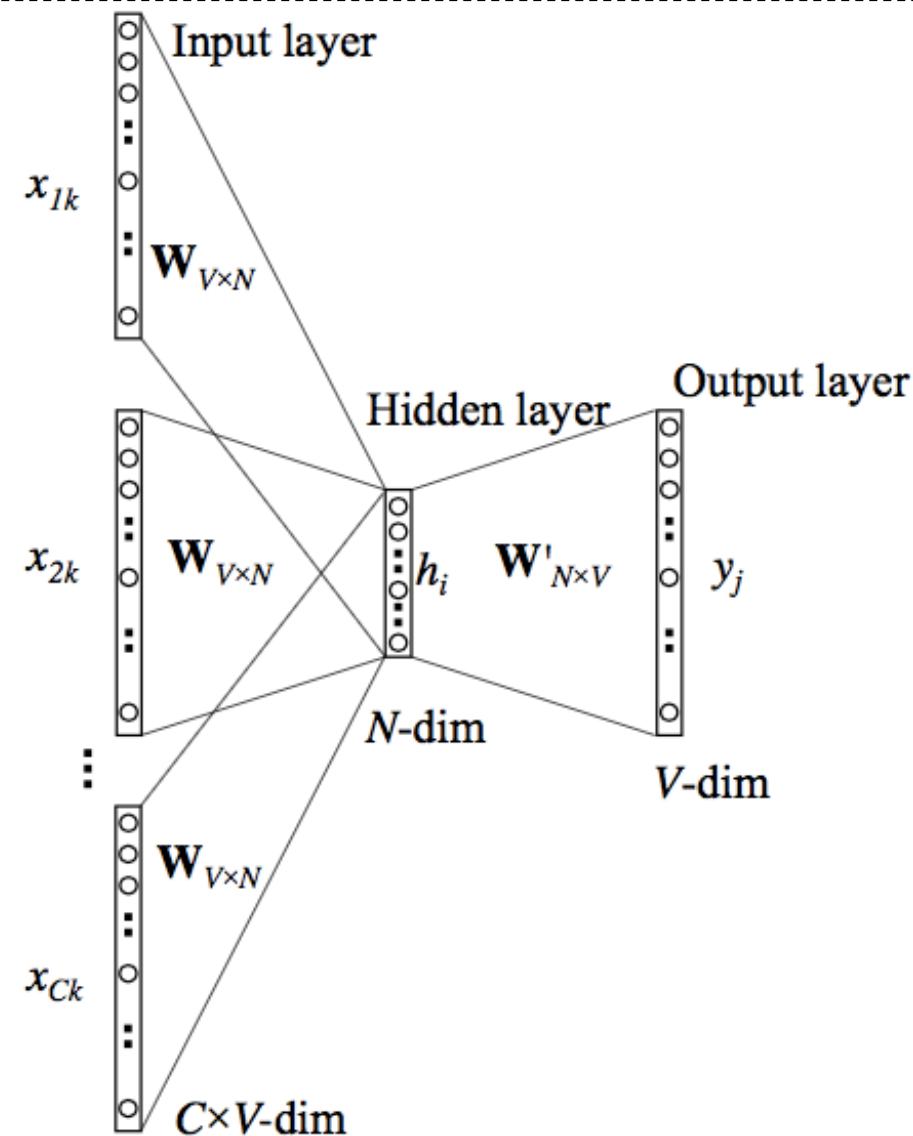
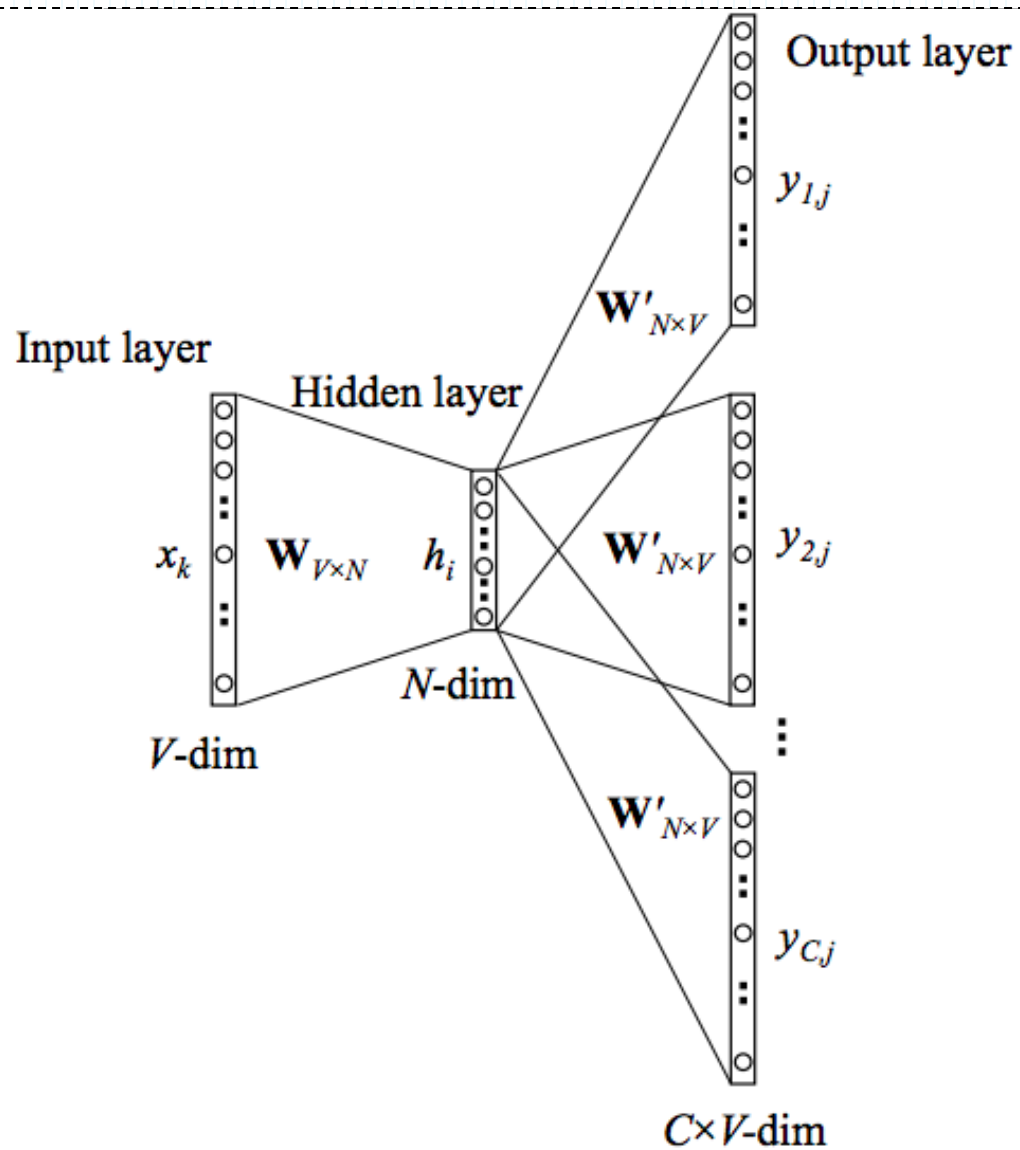
- This idea—“given a word, predict the surrounding words”

she had to pay using ____ instead of any digital or card-based method.

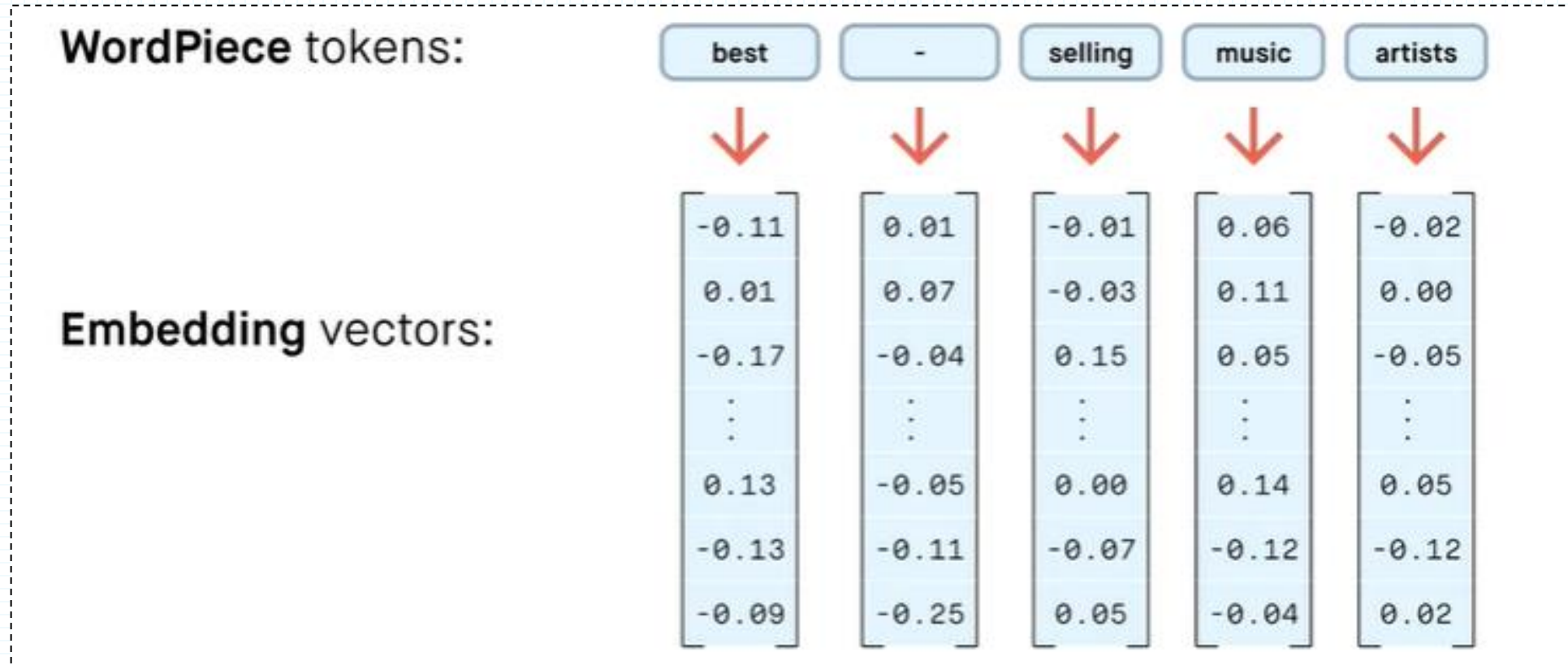


The model is not learning exact word replacement. It is learning a region of meaning.

word2vec



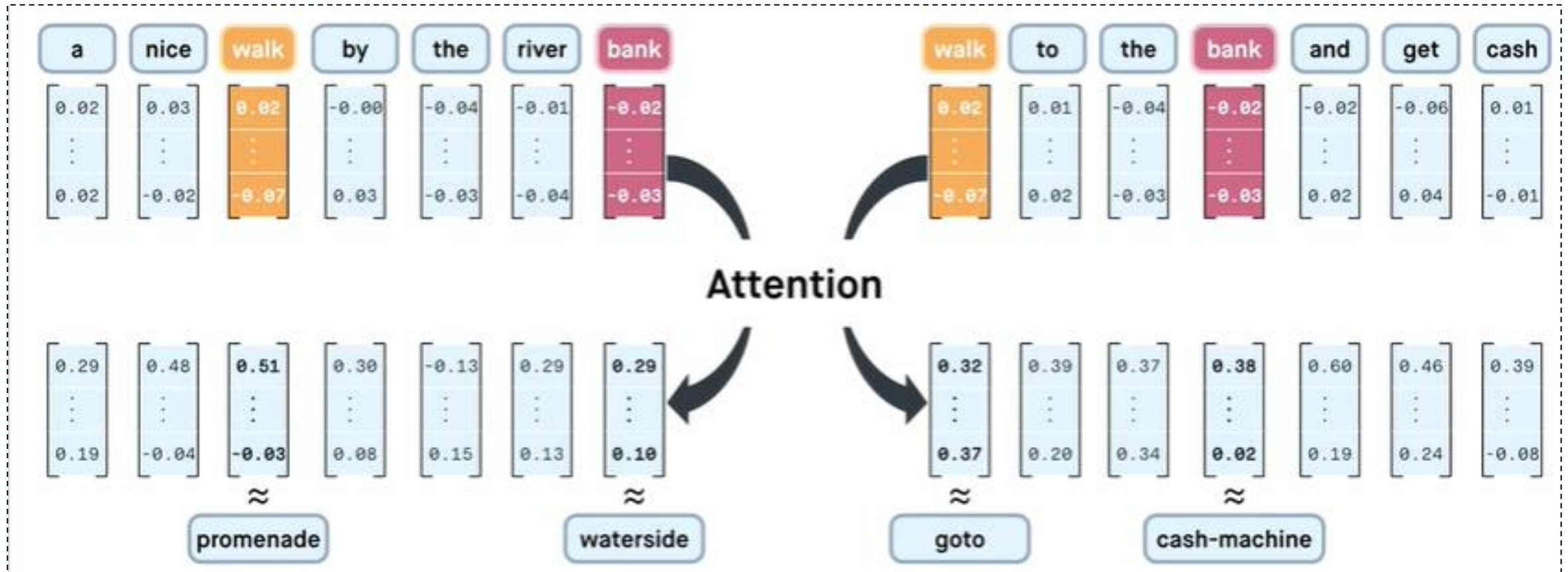
Embedding Vector

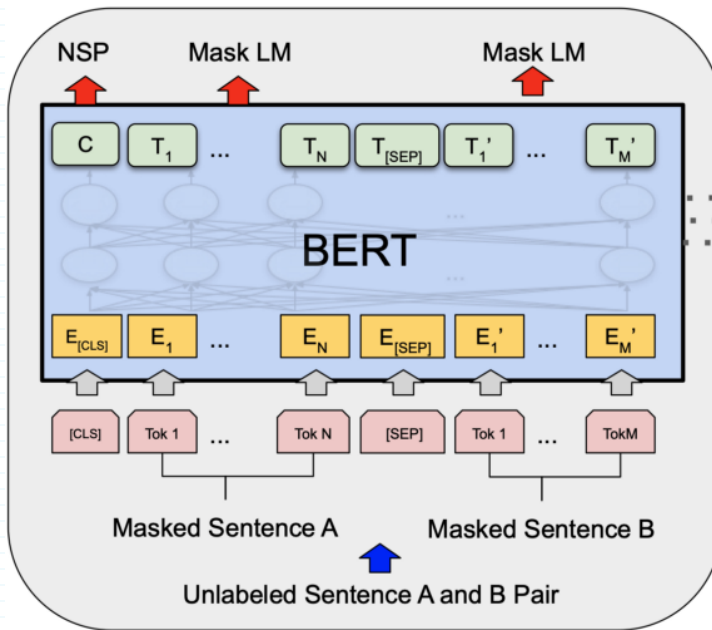


But... Contextual Embedding



Solution.. Attention





Attention..
BERT as an example

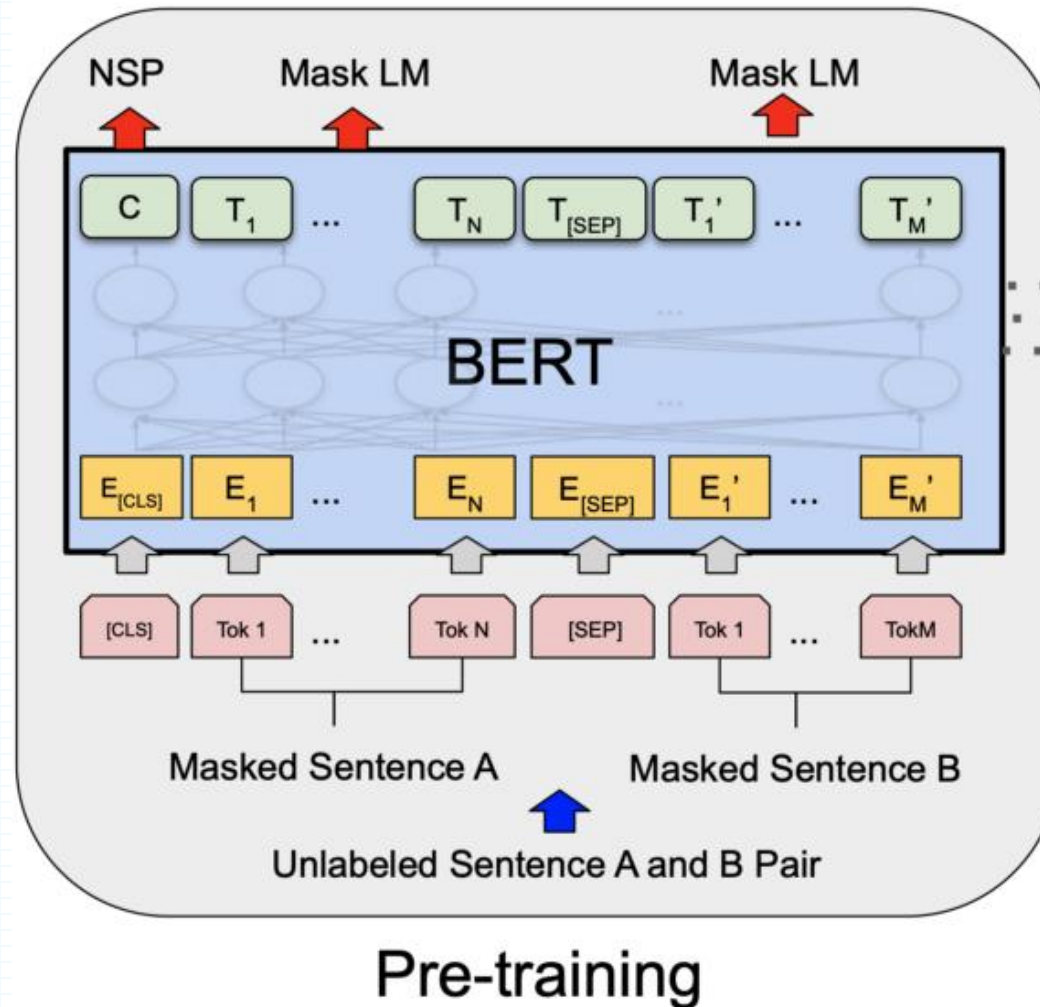
BERT

Next Sentence
Prediction

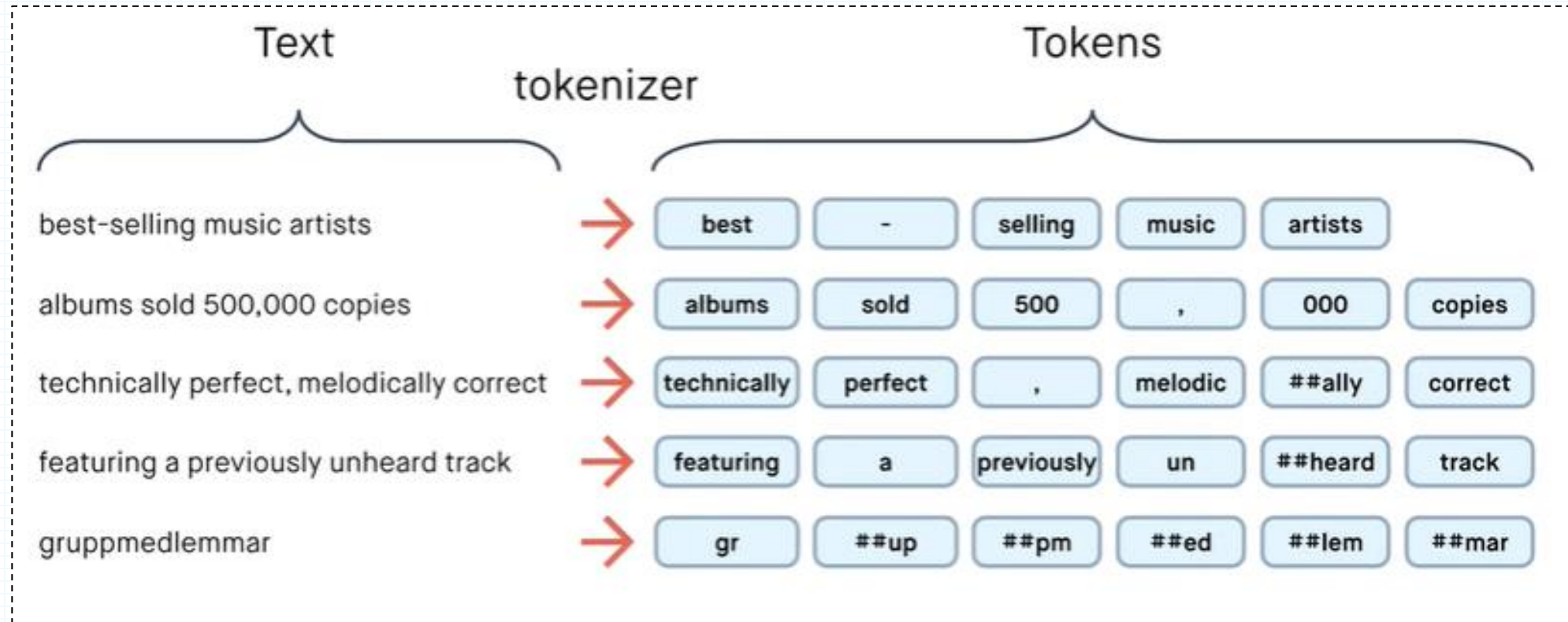
BERT: Pre-training of Deep Bidirectional Transformers for
Language Understanding

2018

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



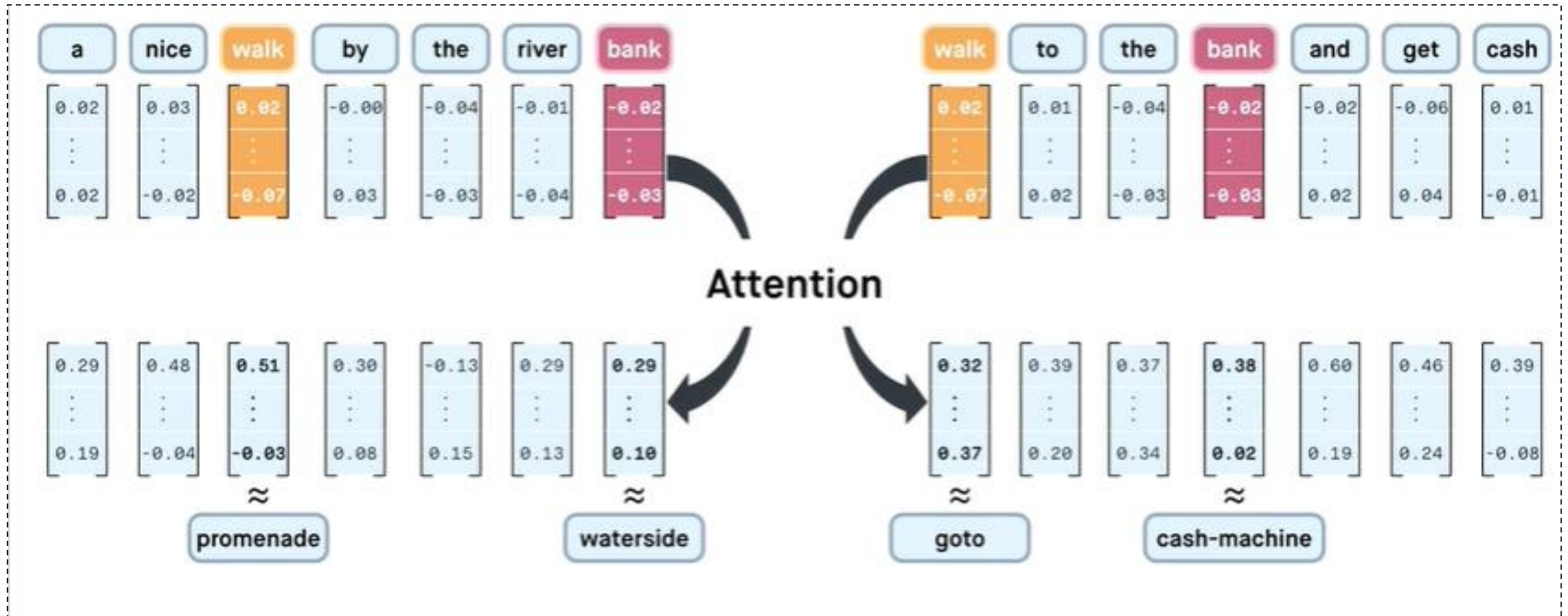
Tokenization



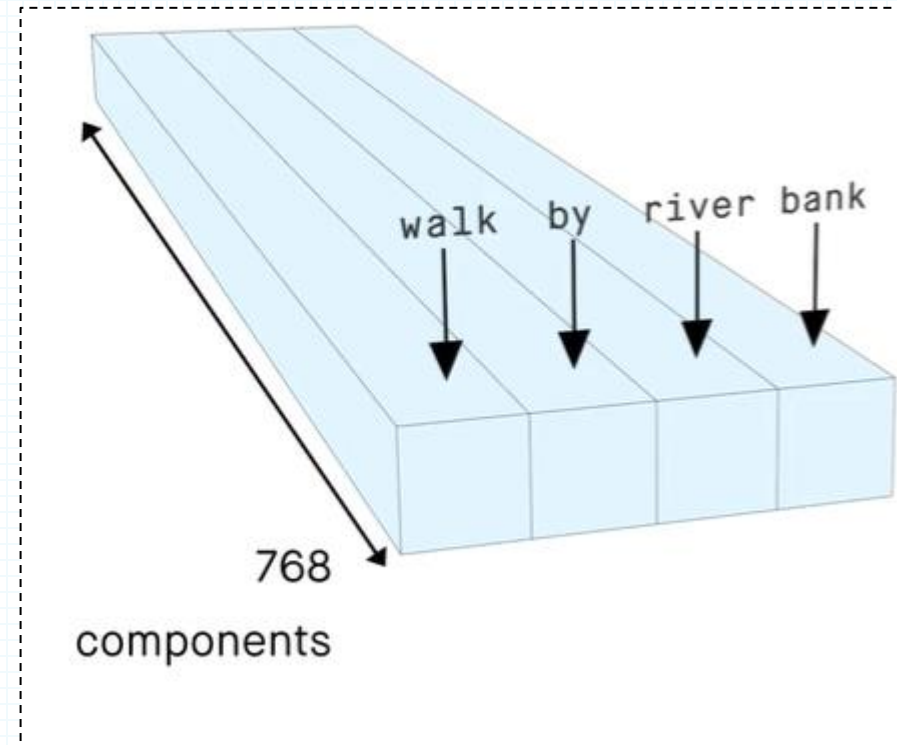
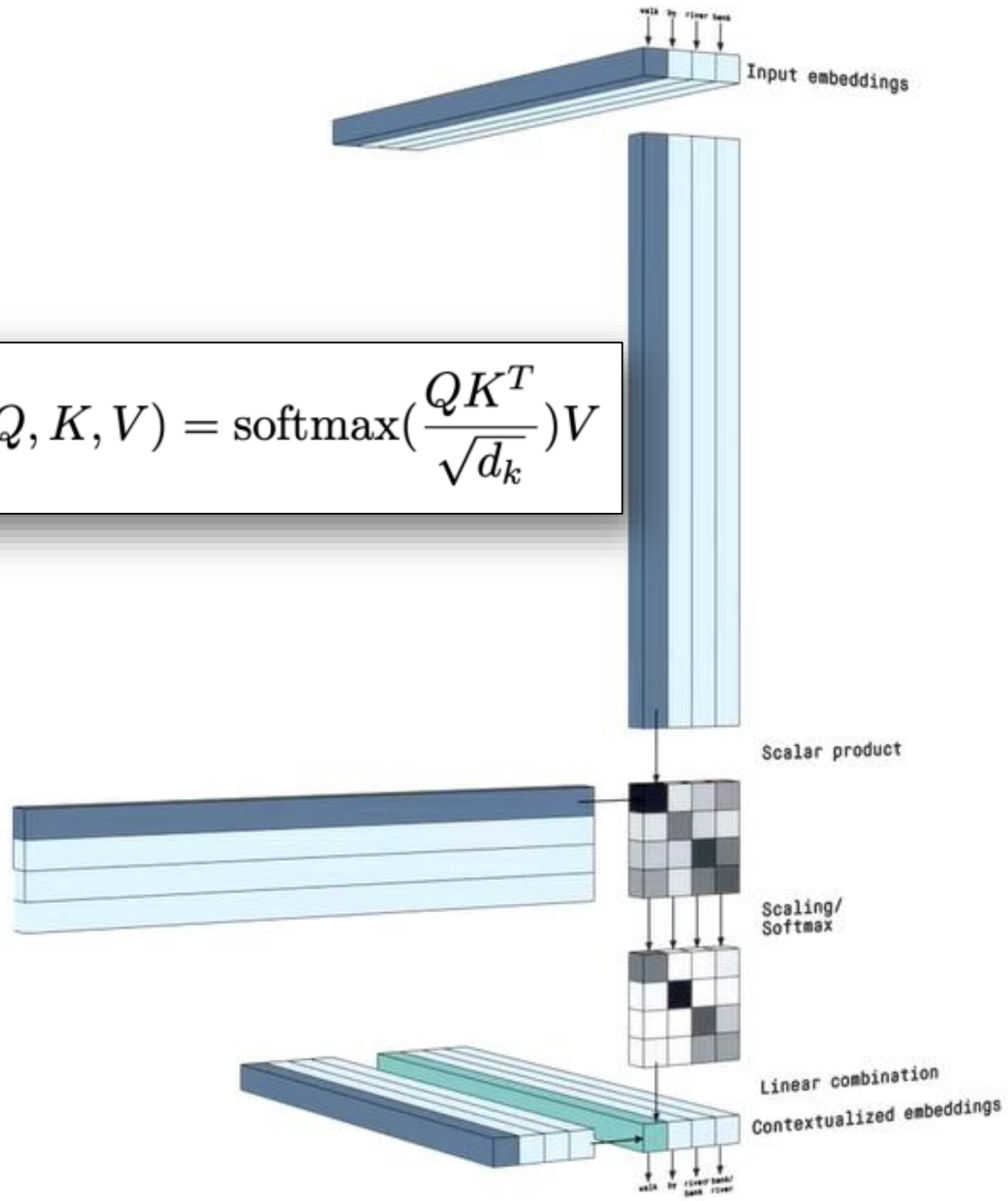
Initial Embedding



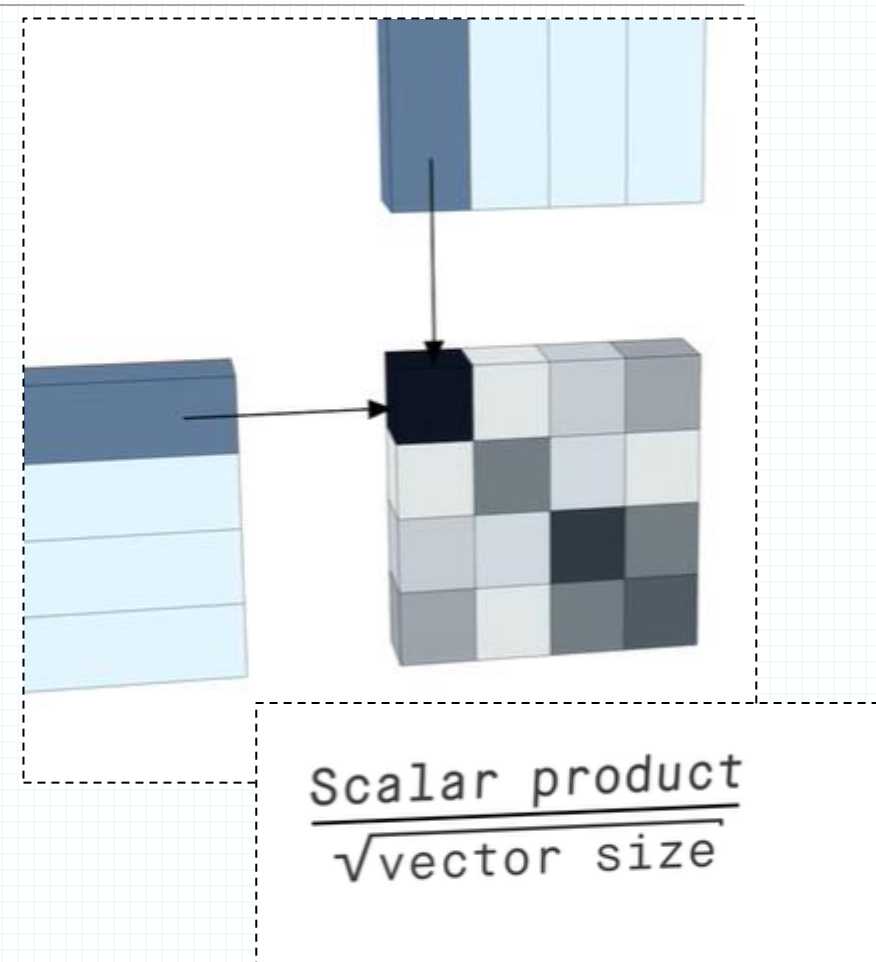
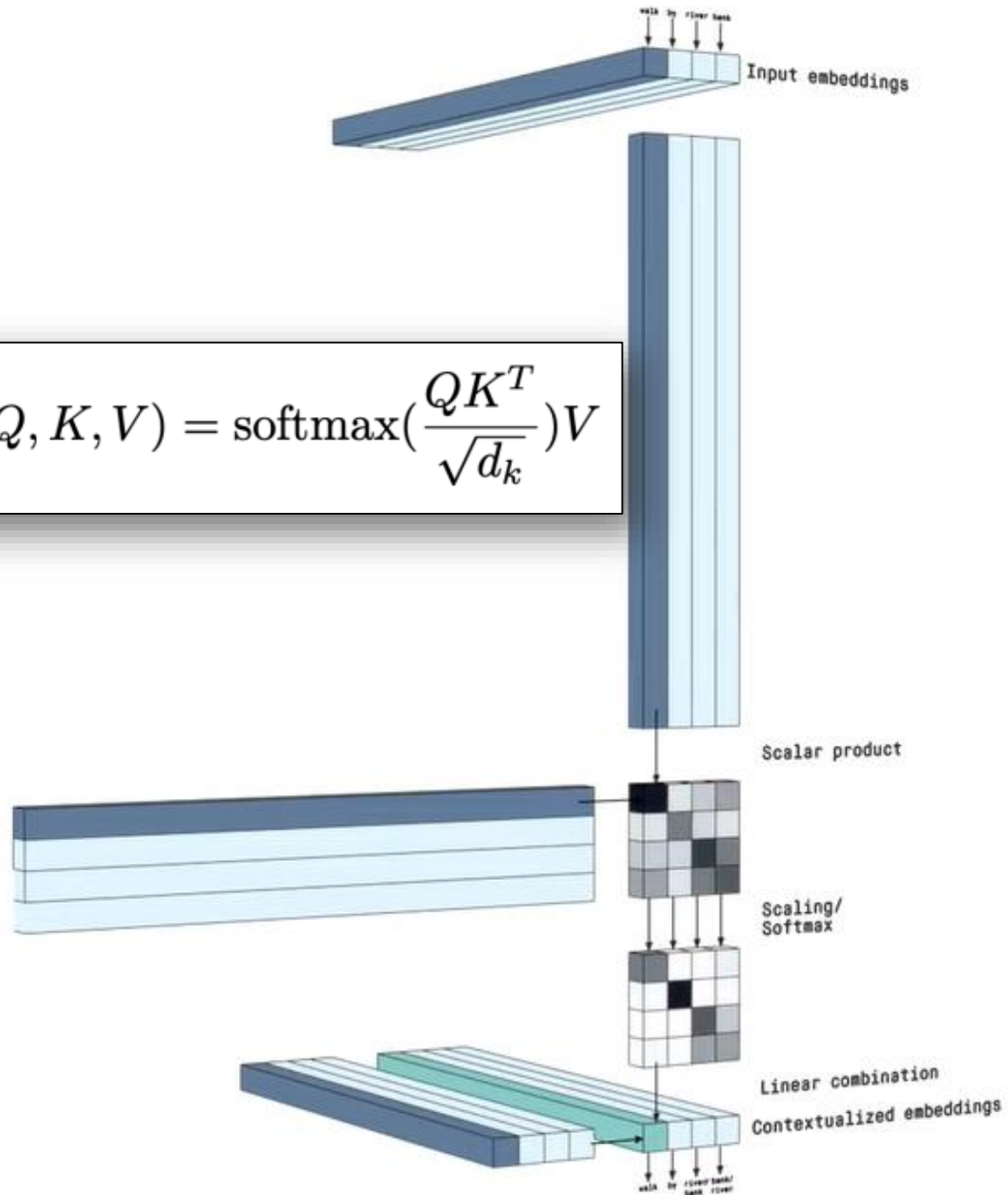
Contextual Embedding



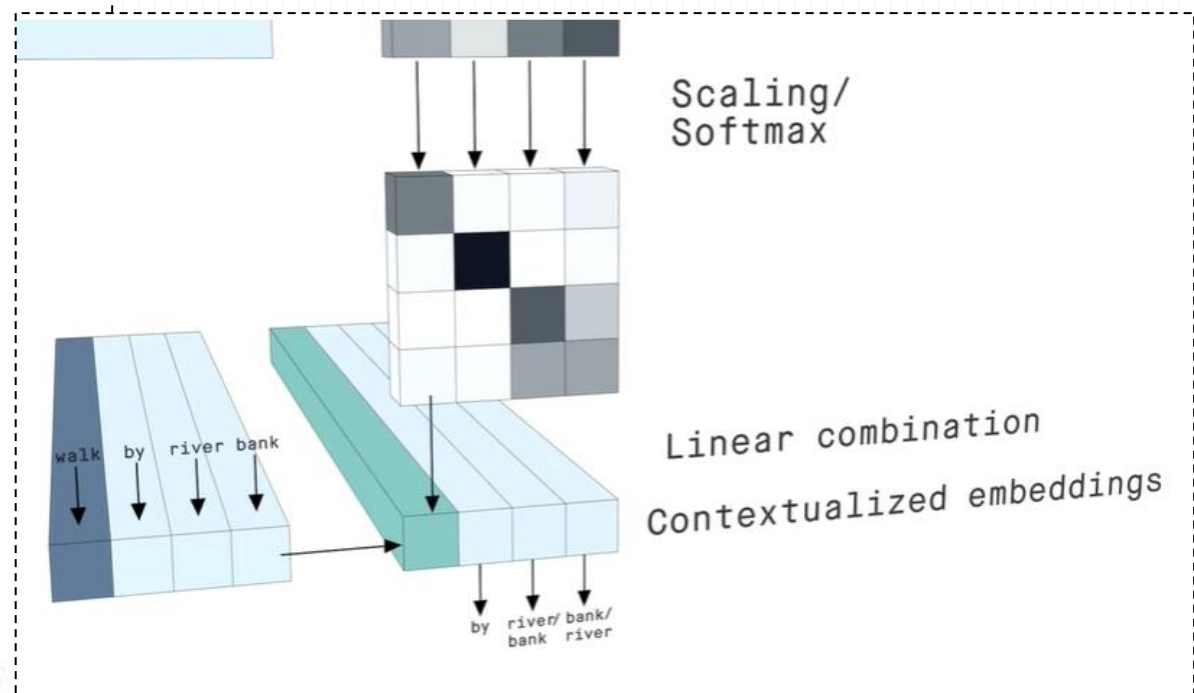
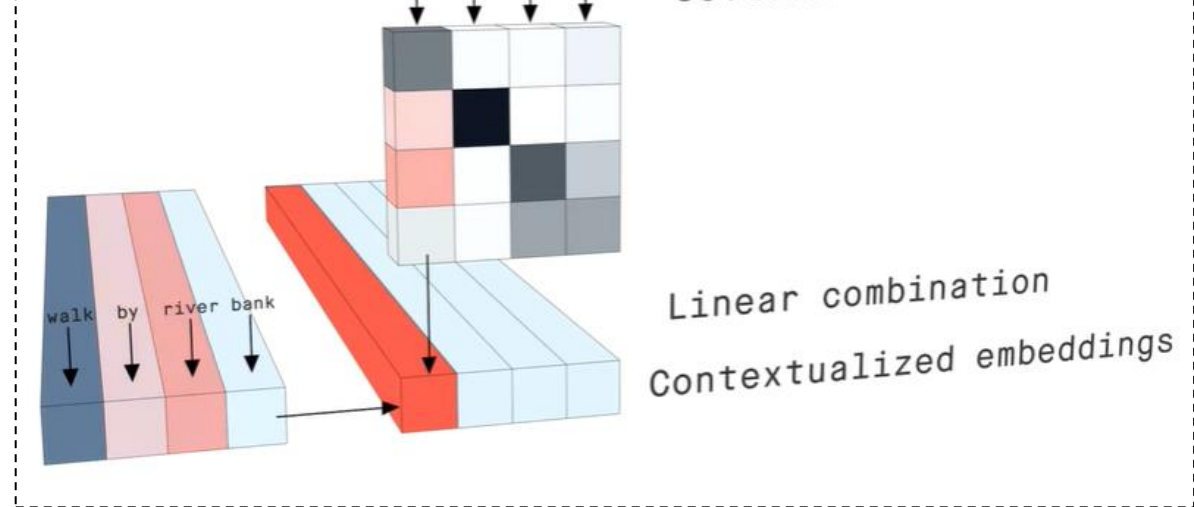
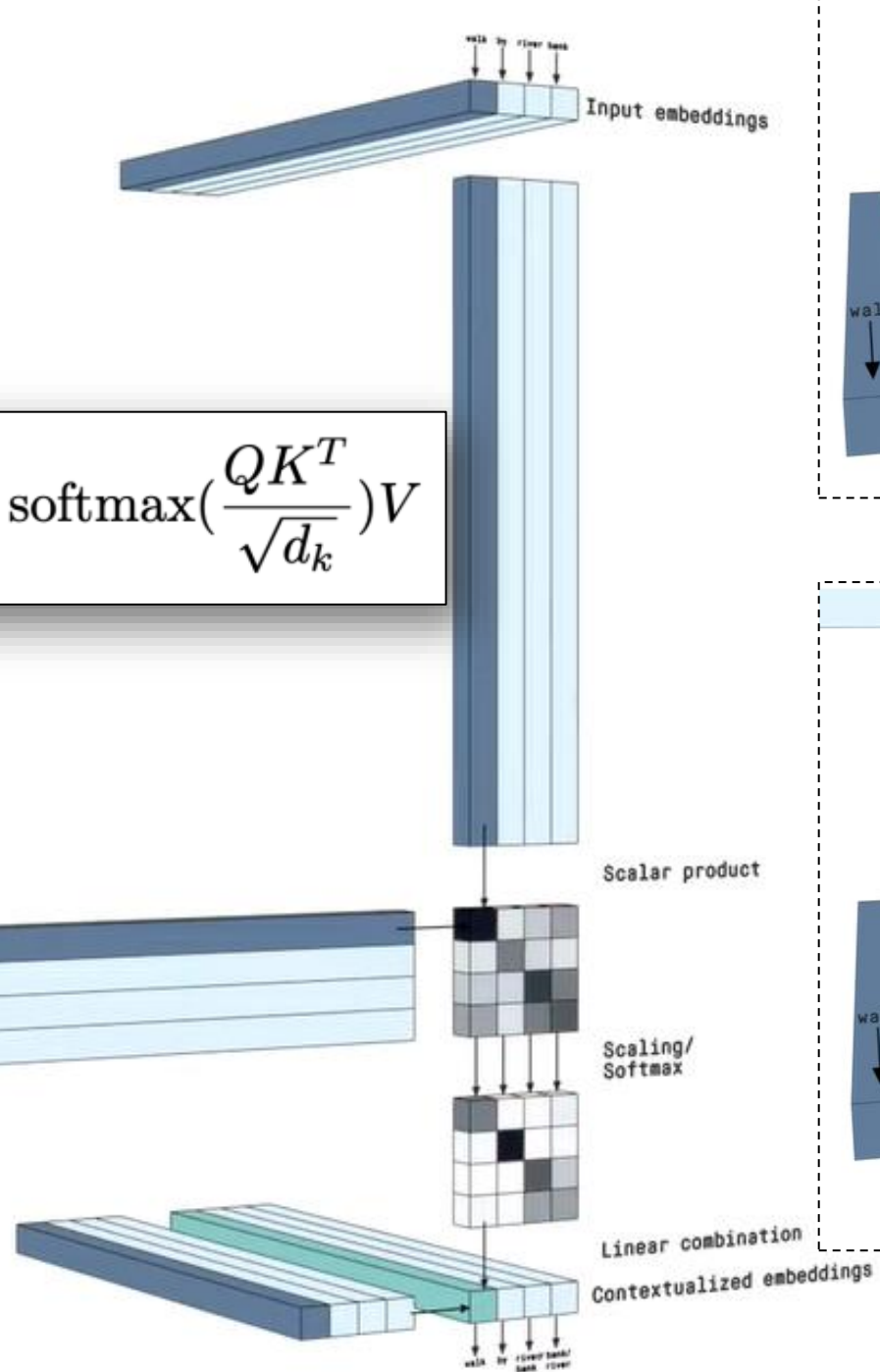
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



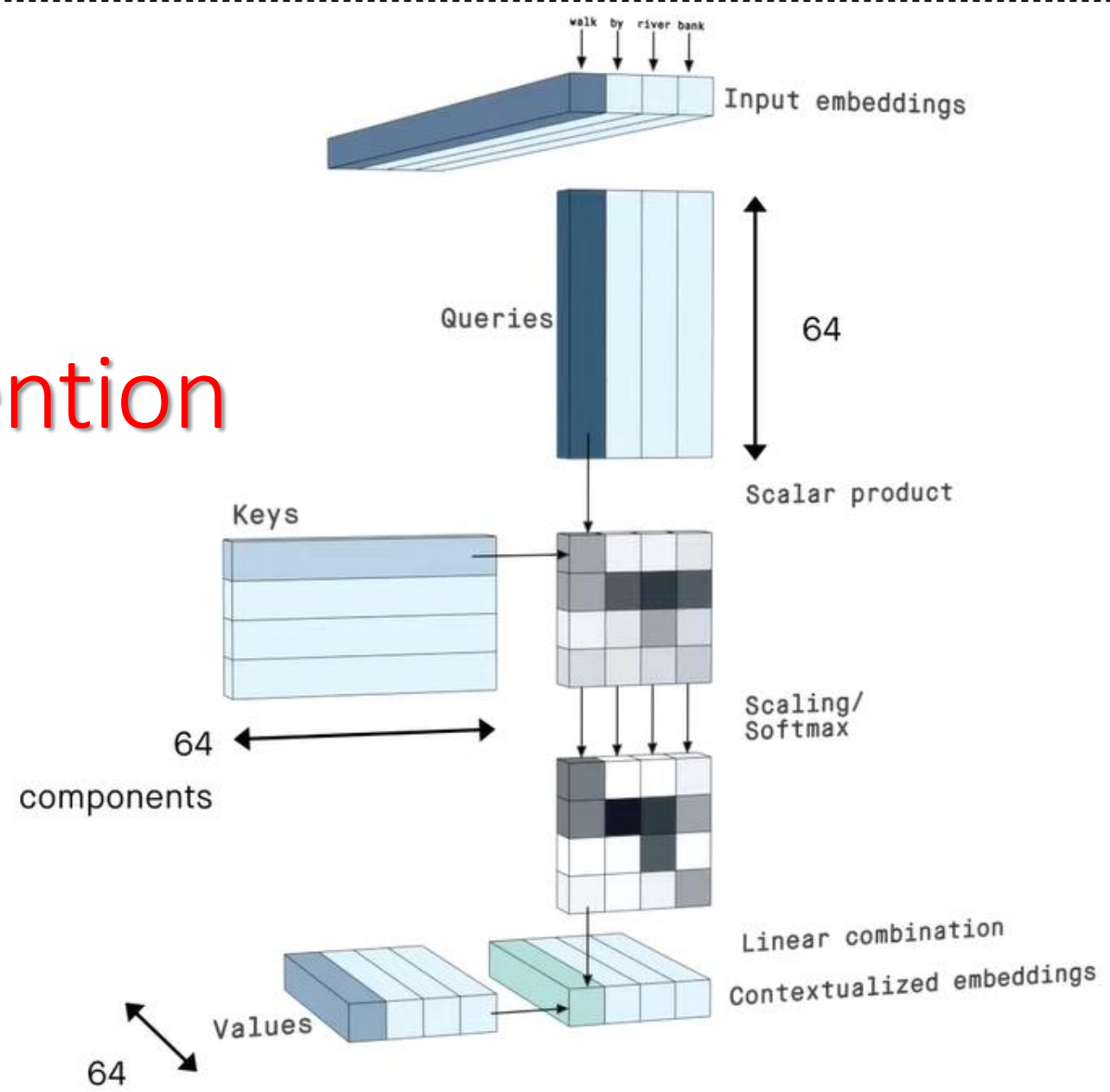
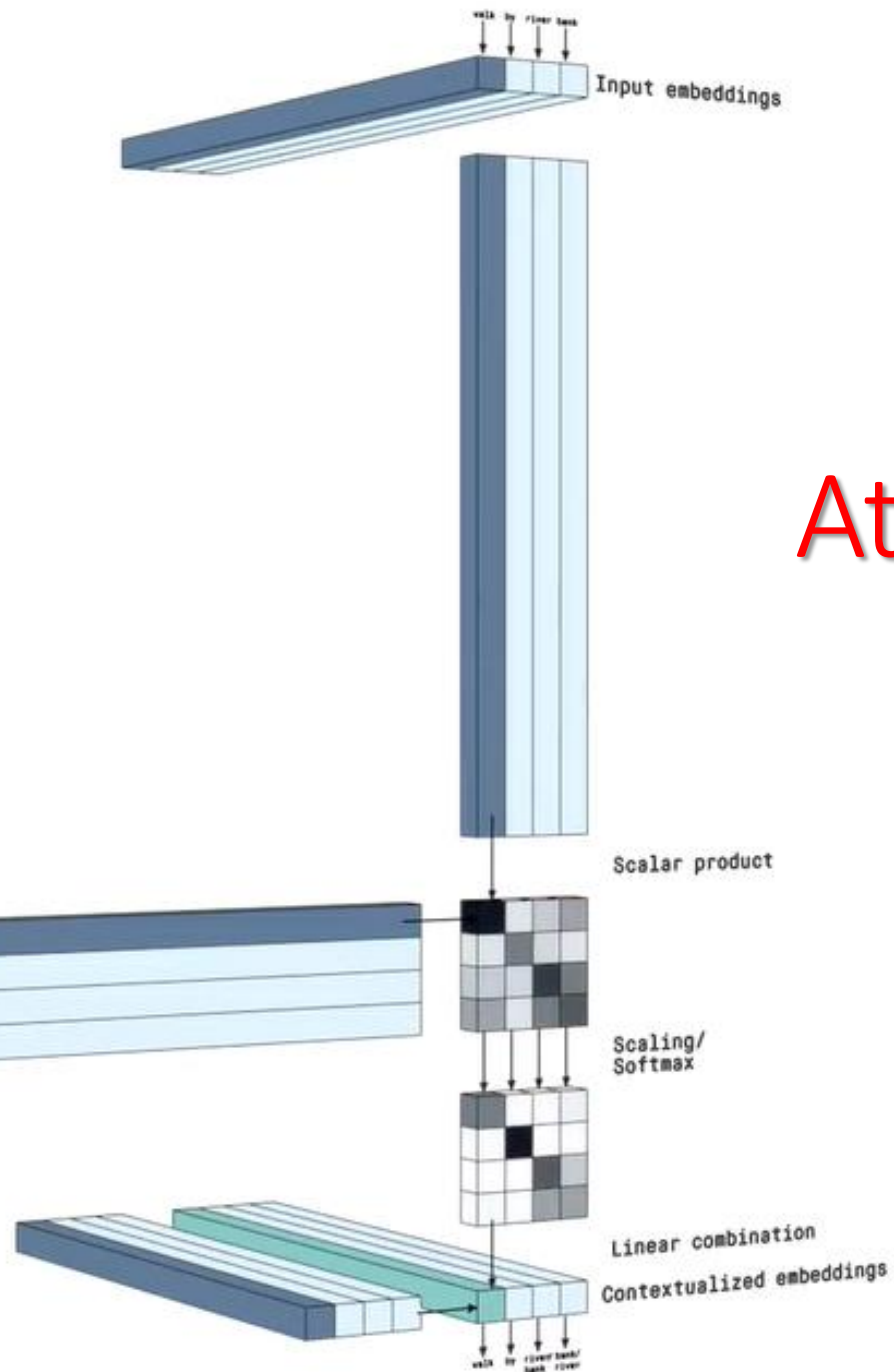
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



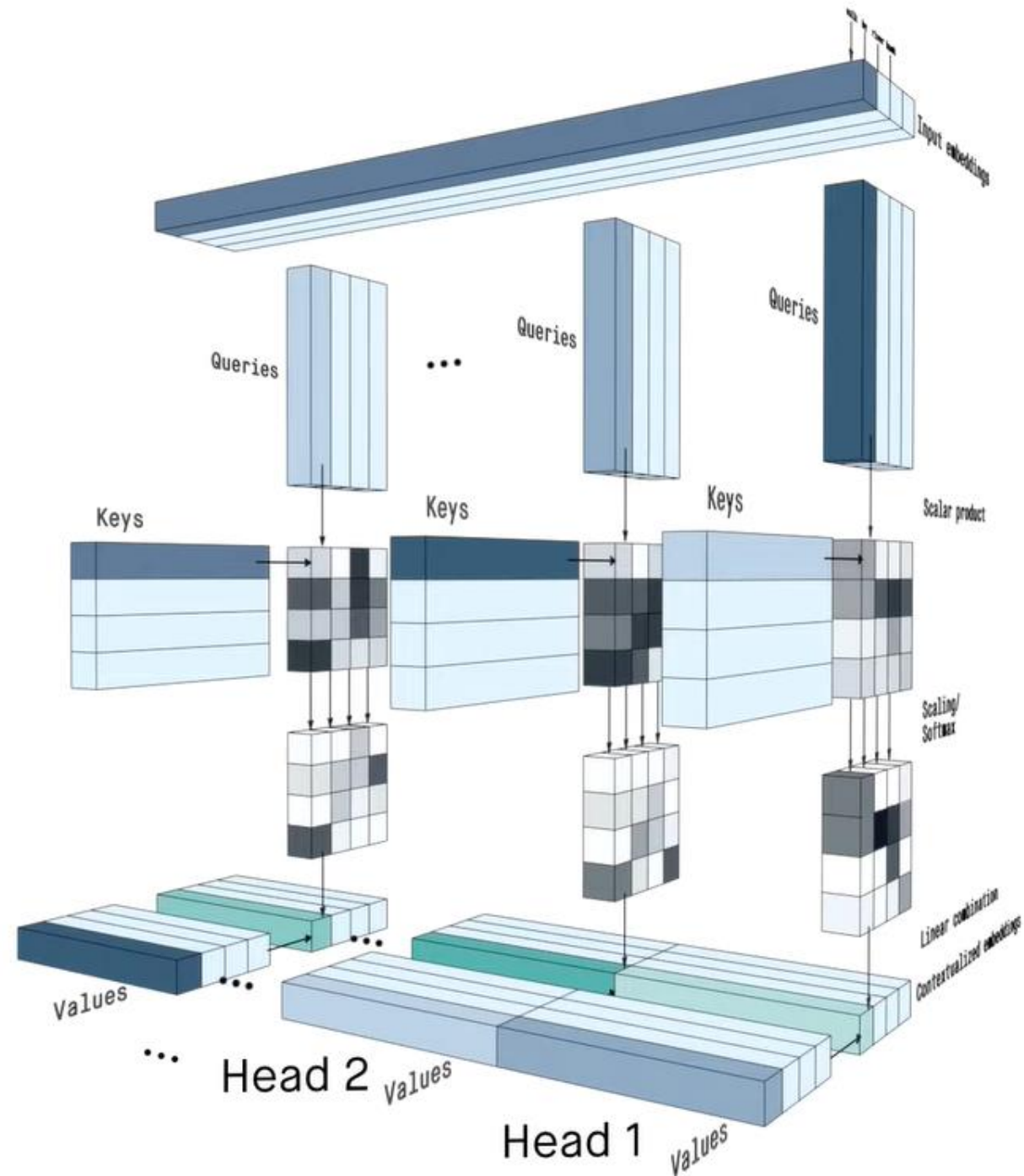
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Attention



Multi-head attention



Enrich the input!

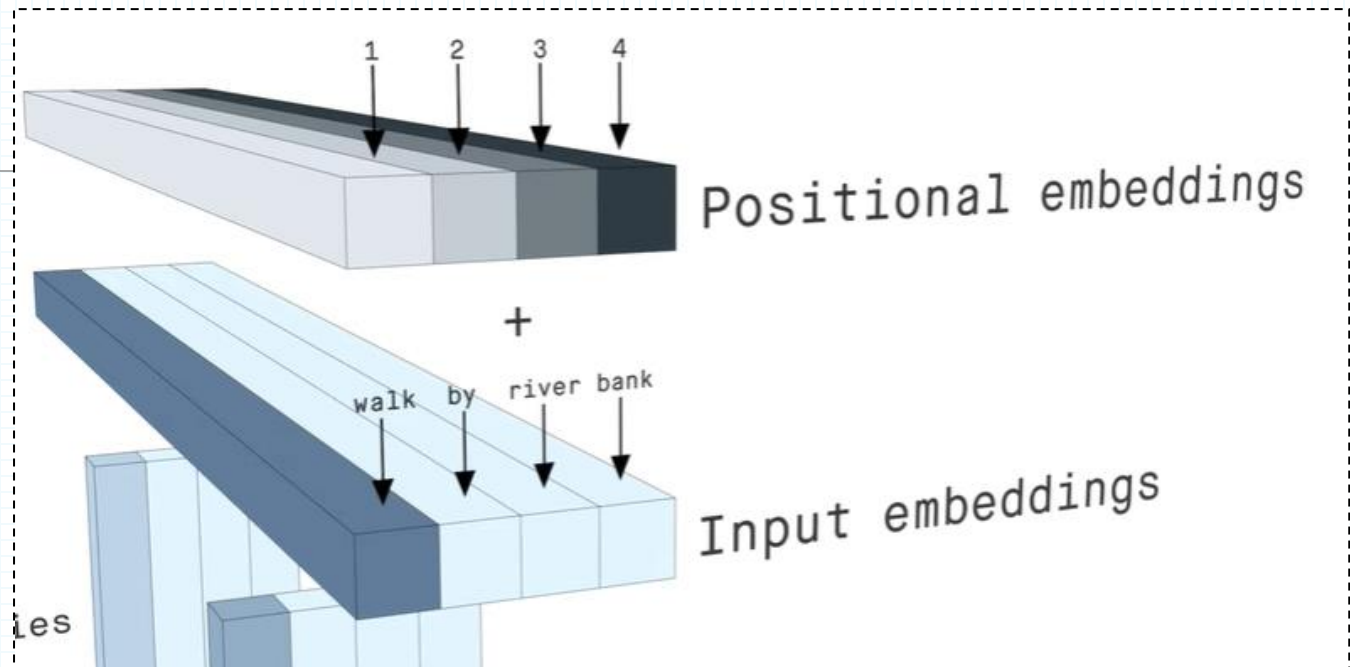
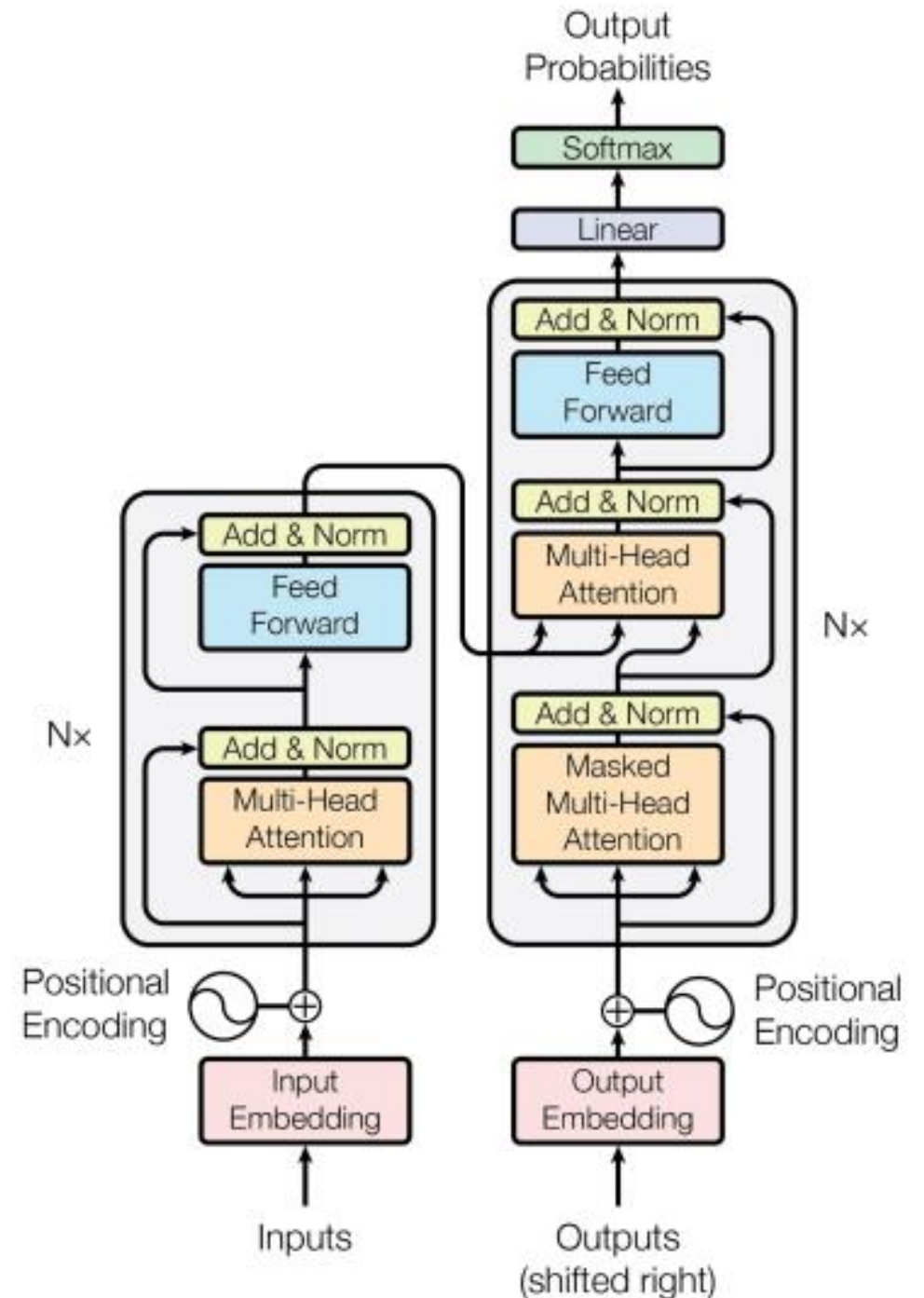


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Attention... and Transformers

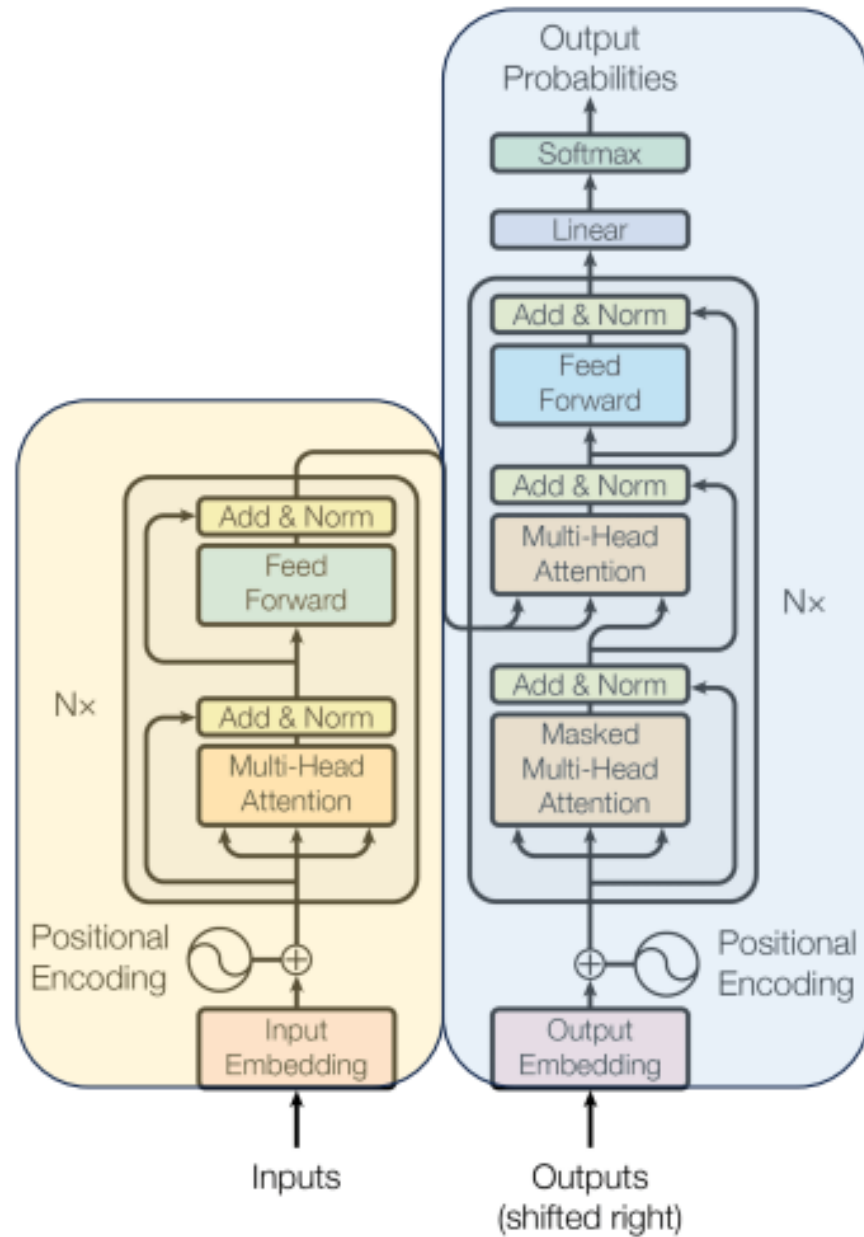
Transformers

- 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)



BERT
Oct 2018

Representation

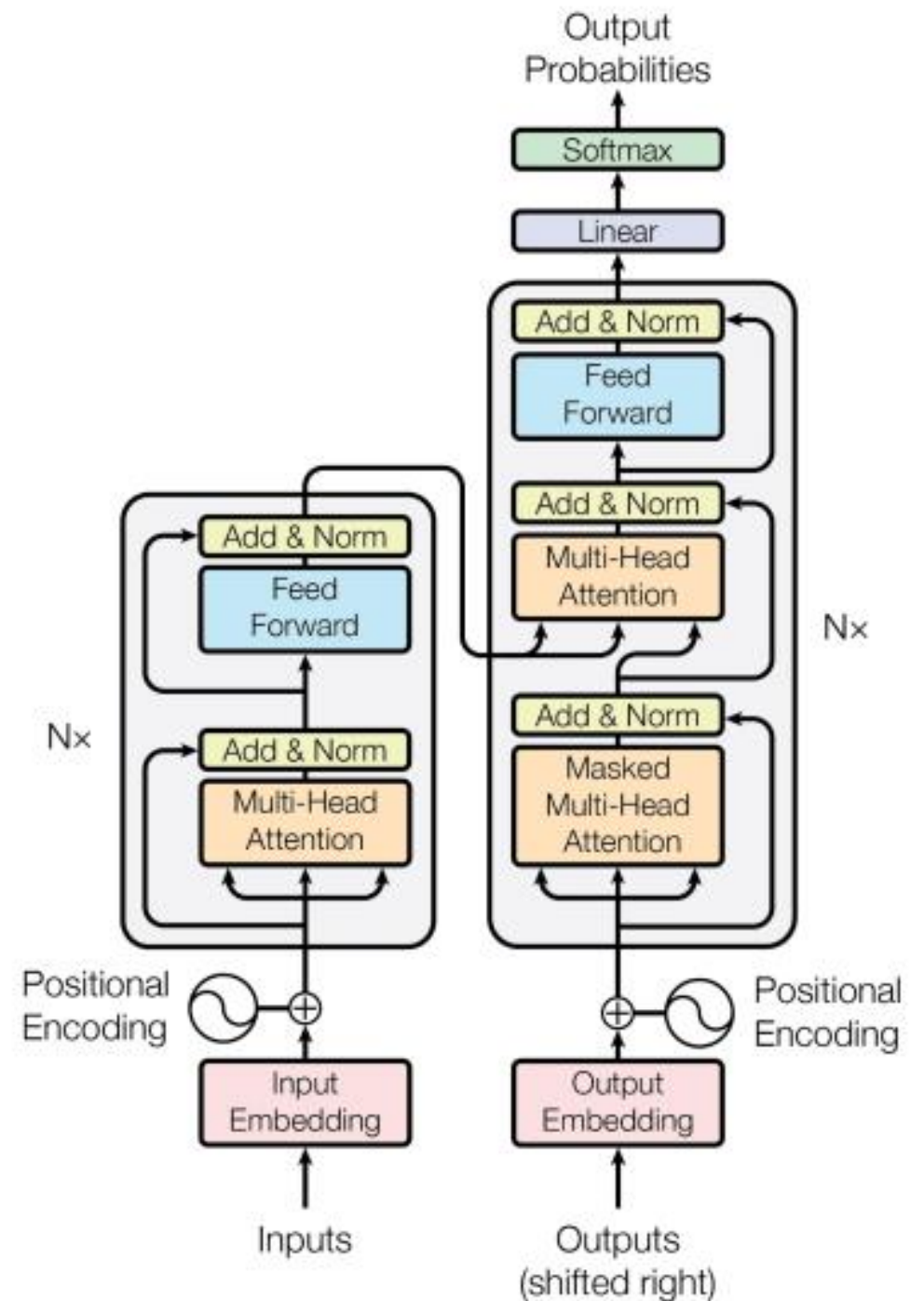
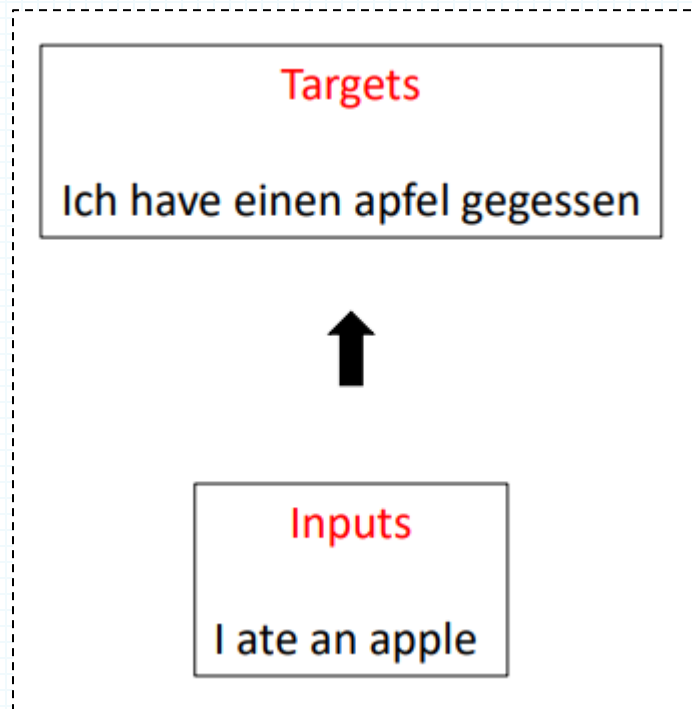


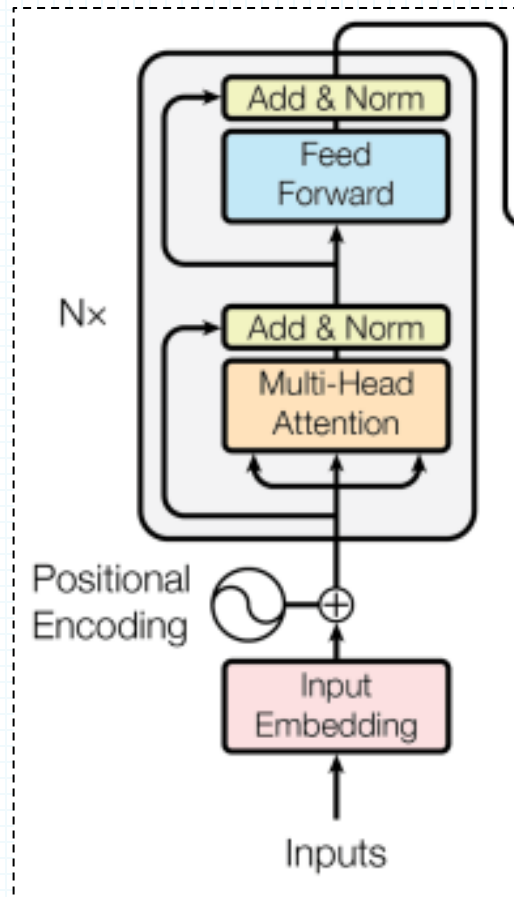
GPT
Jun 2018

Generation

Transformers

- Example: Machine Translation

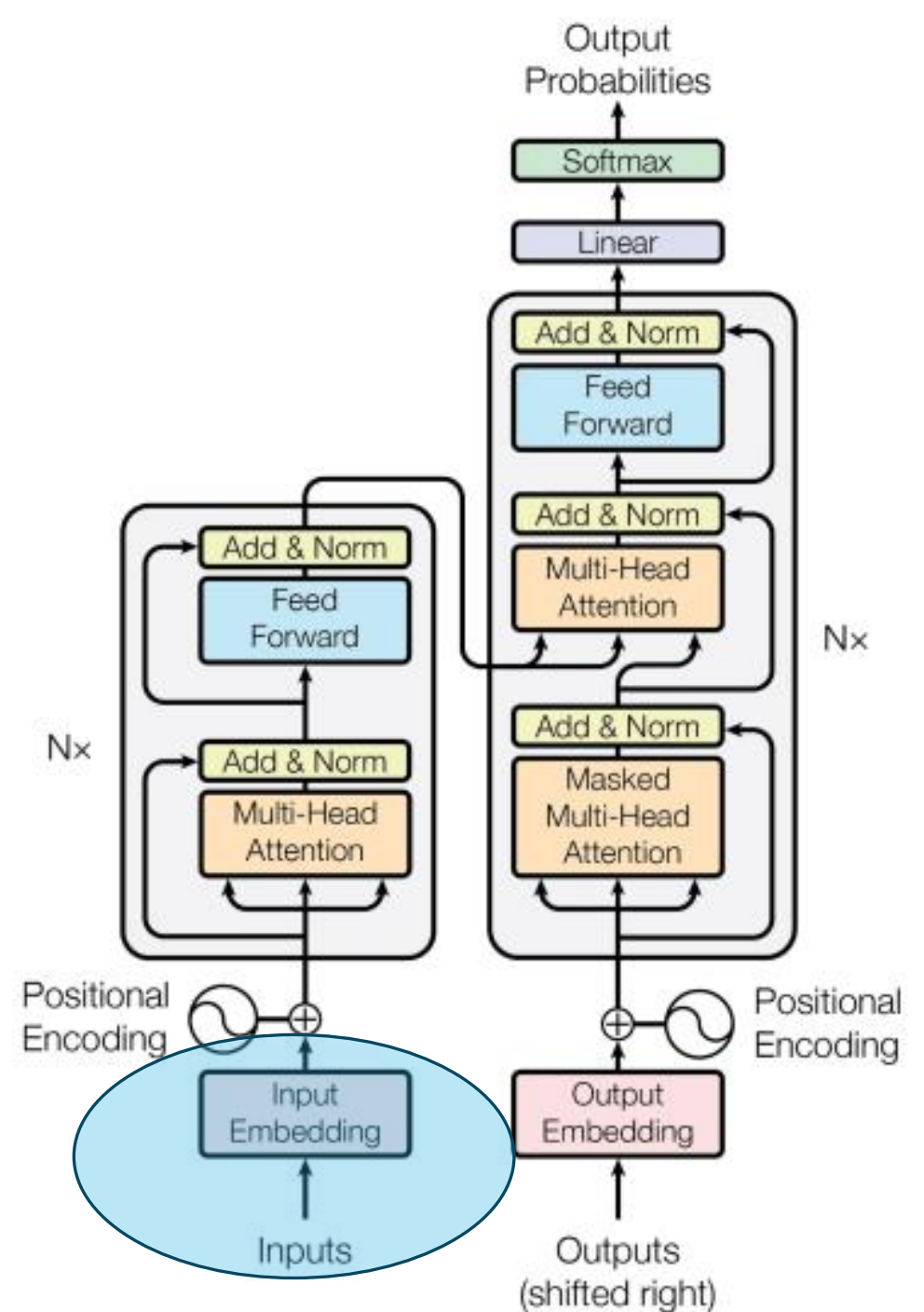
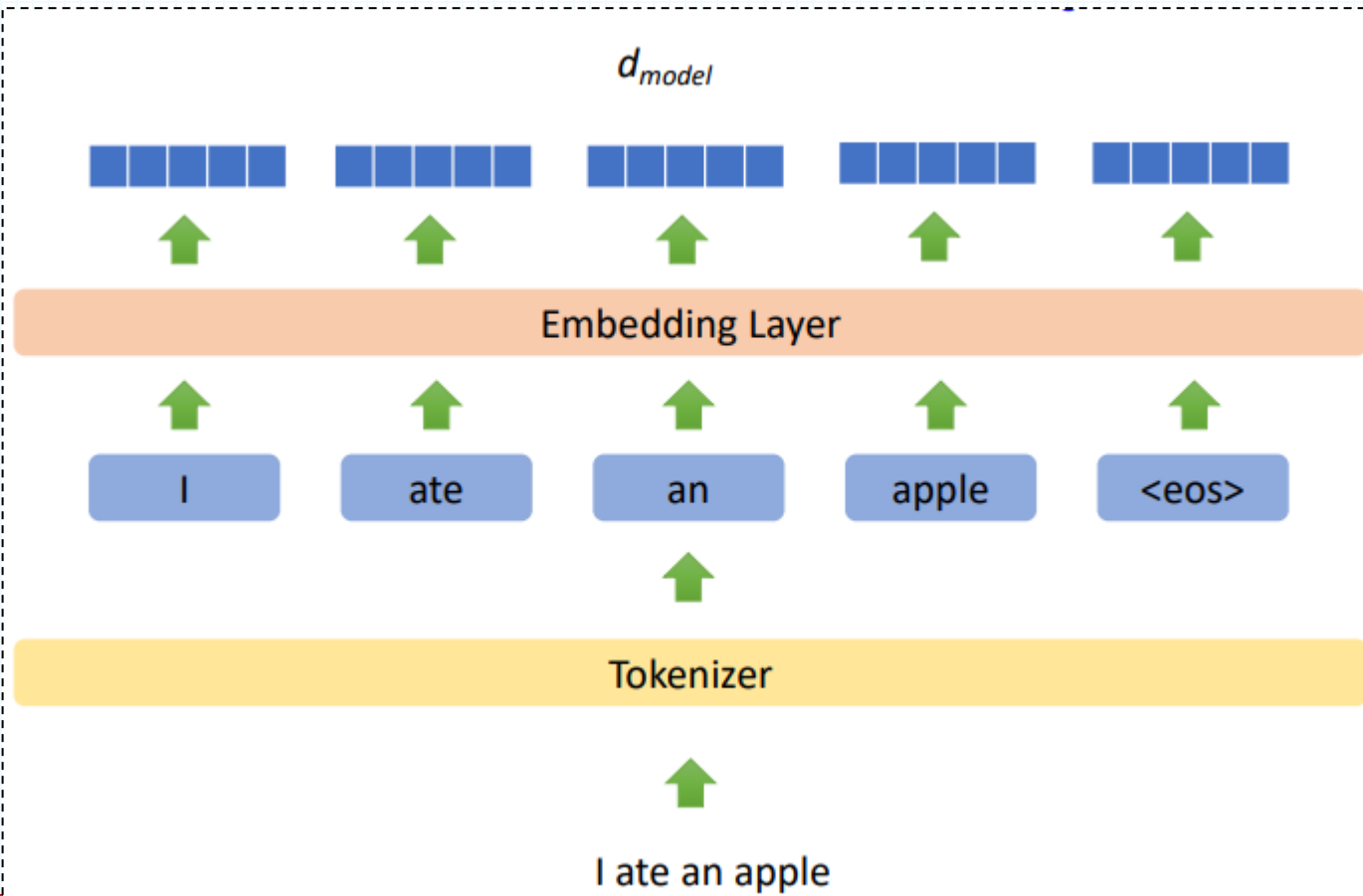




Encoder Part

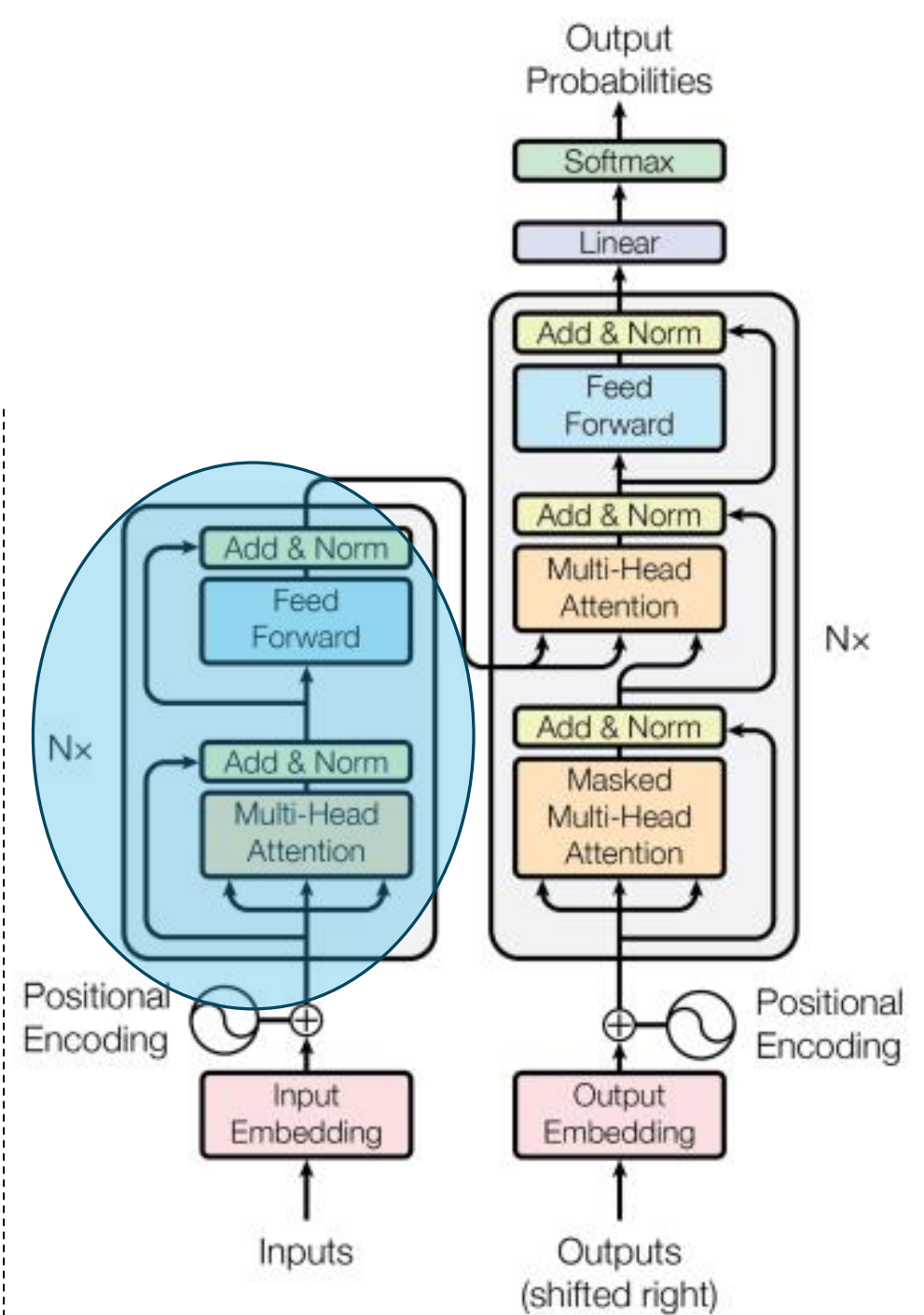
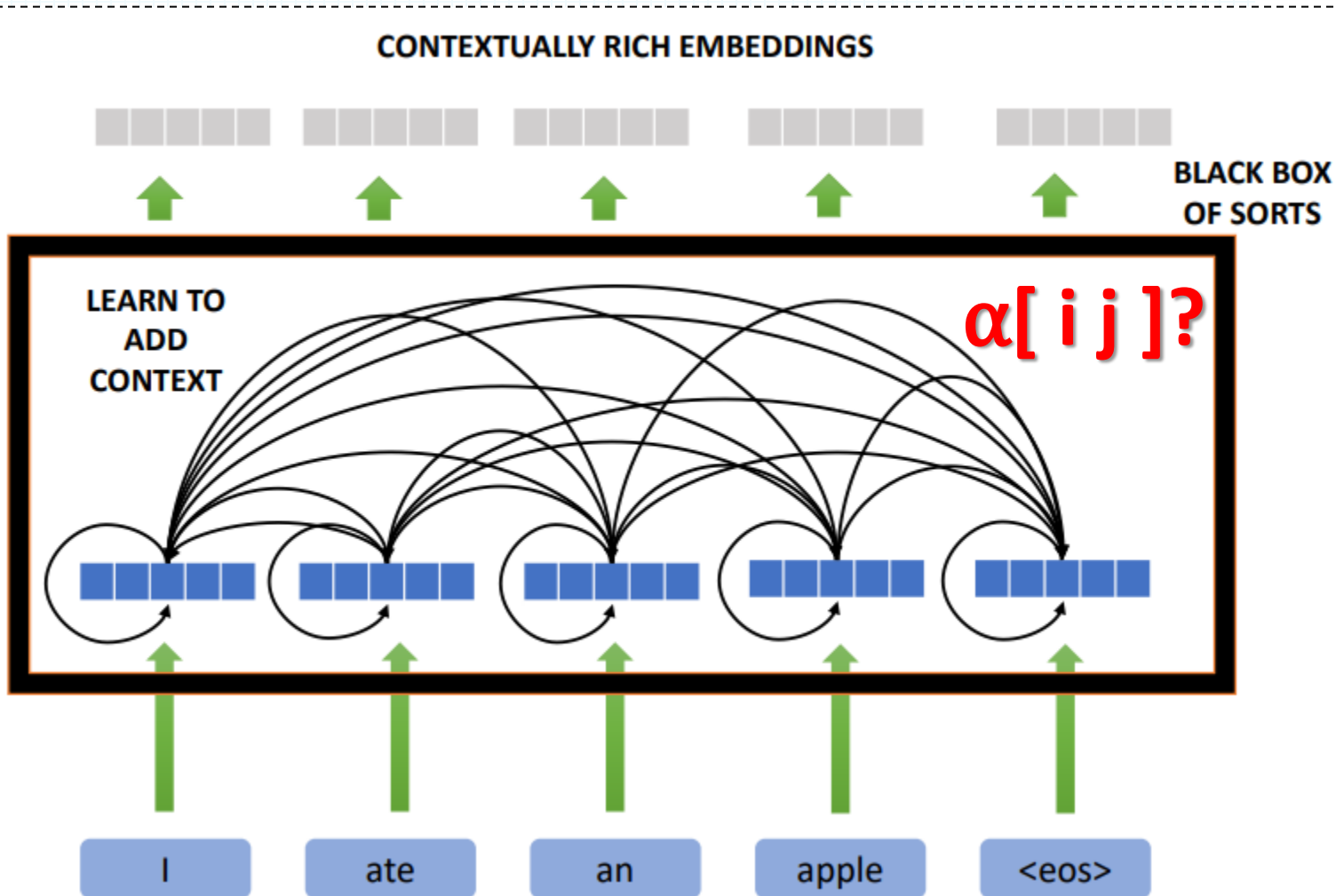
Transformers

- Processing Input



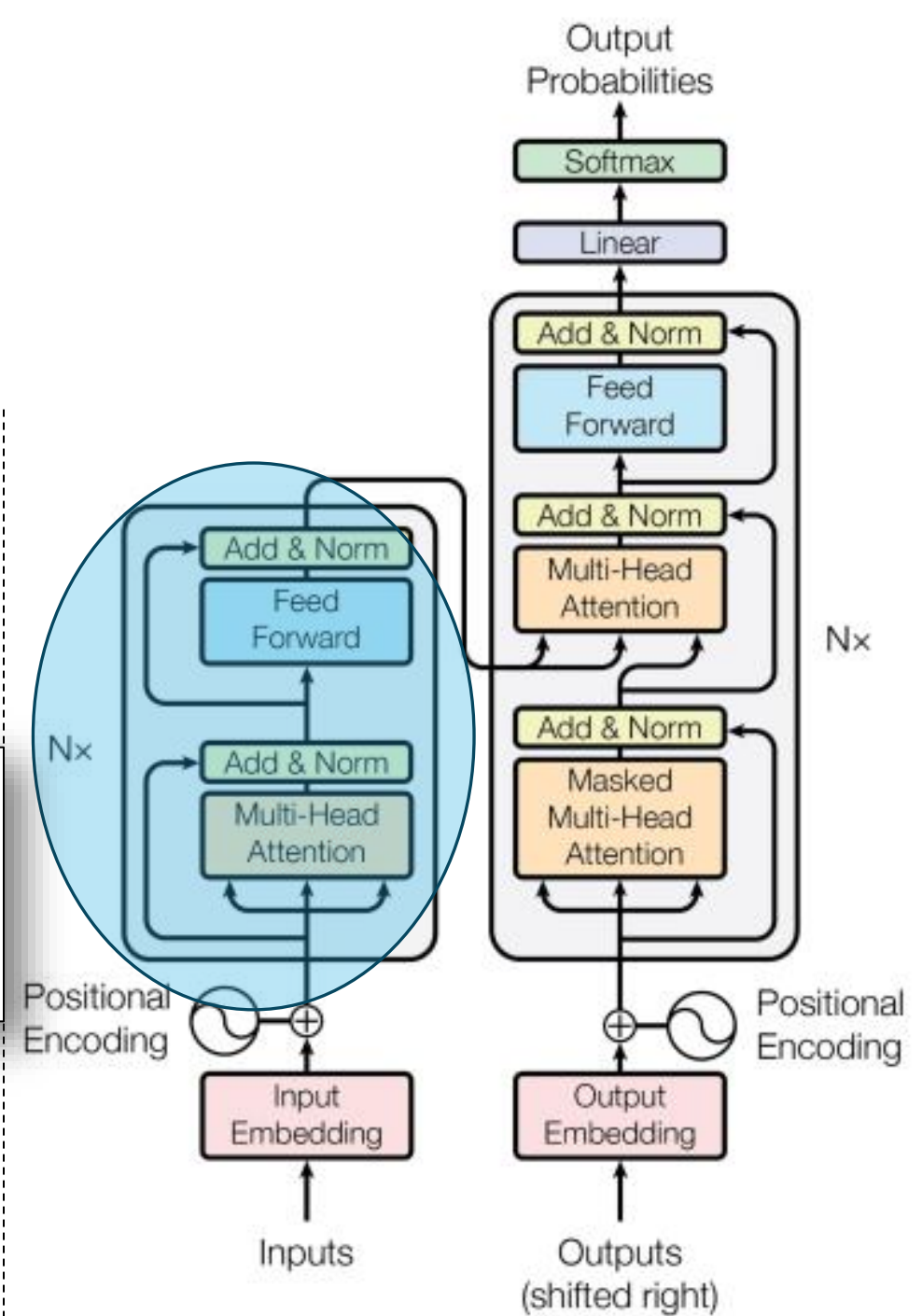
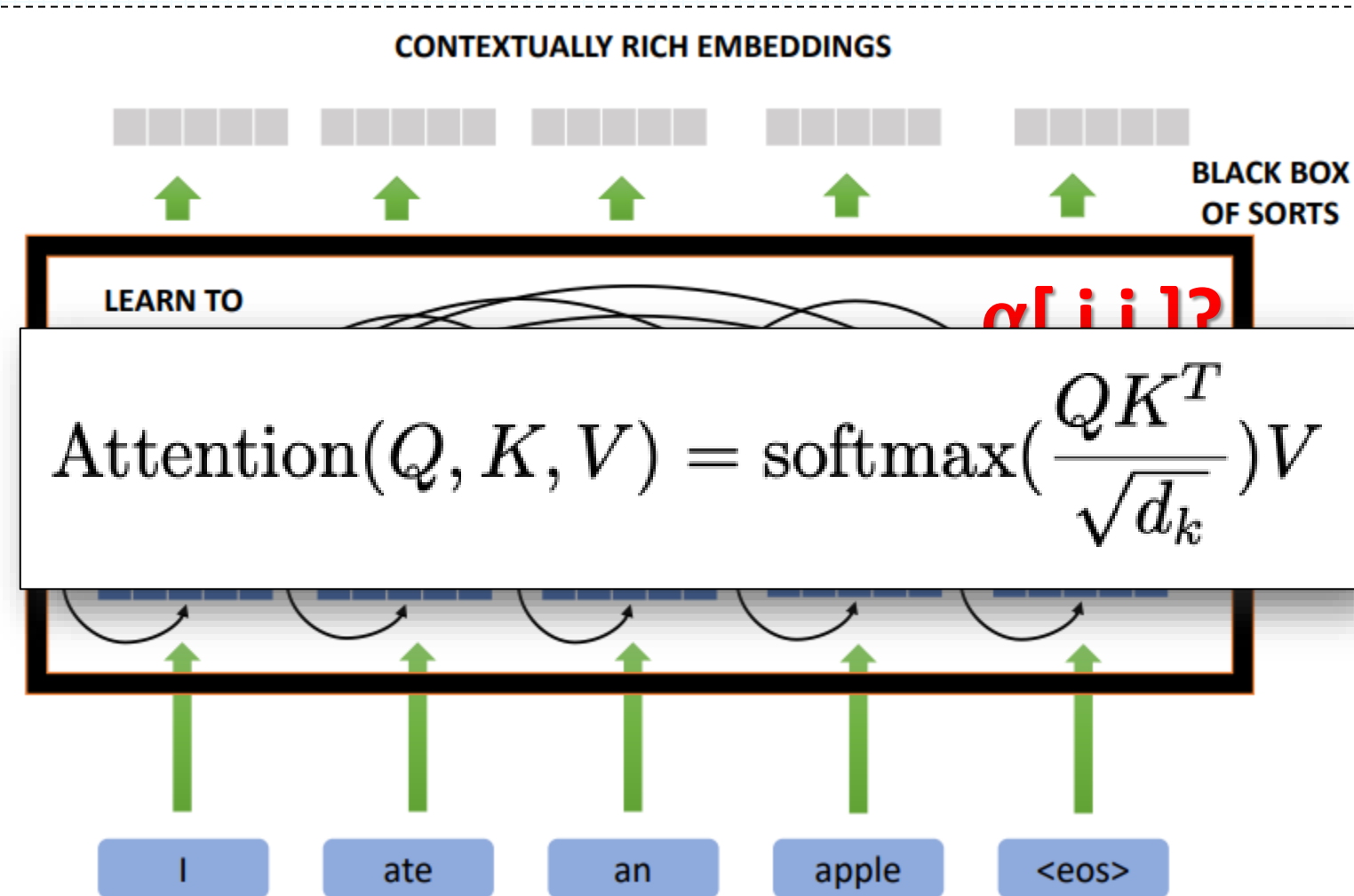
Transformers

- Learn to add context



Transformers

- Learn to add context



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 index of the tokens in the context)

- In the attention mechanism, we use an analogy to Information Retrieval

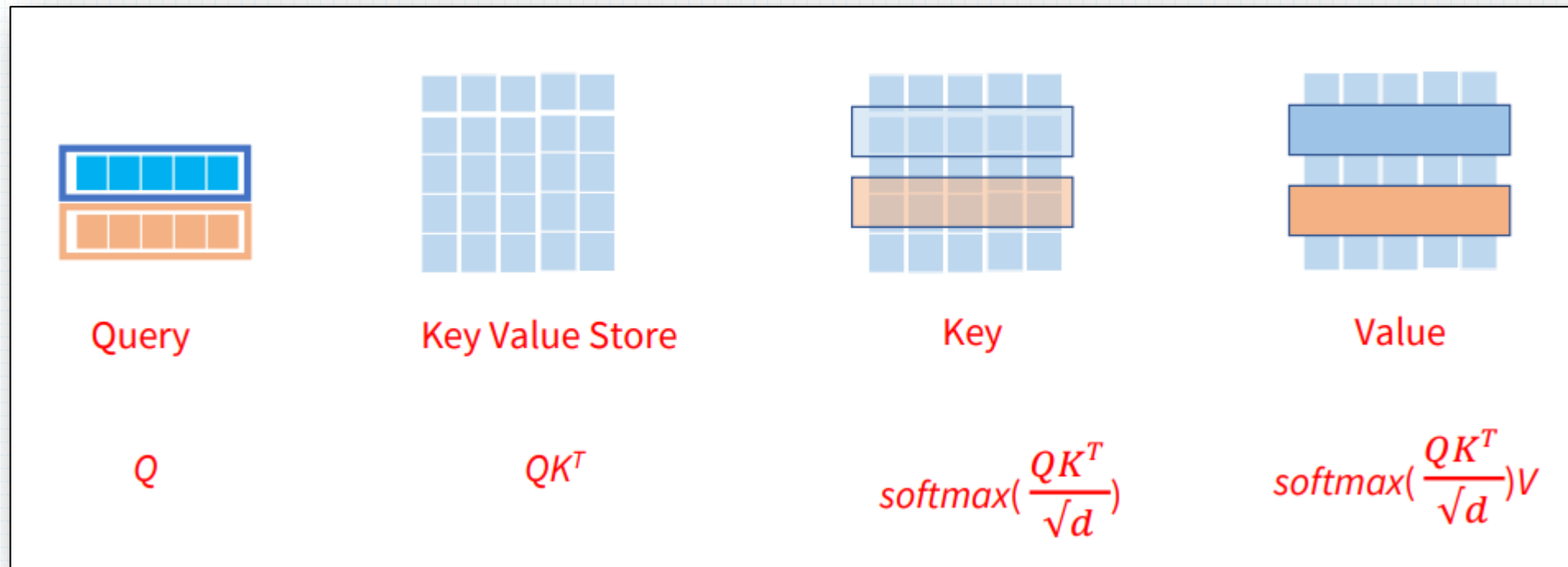
Attention allows the model to dynamically retrieve relevant information (values) from the context, based on how similar other tokens (keys) are to the current focus (query).

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.

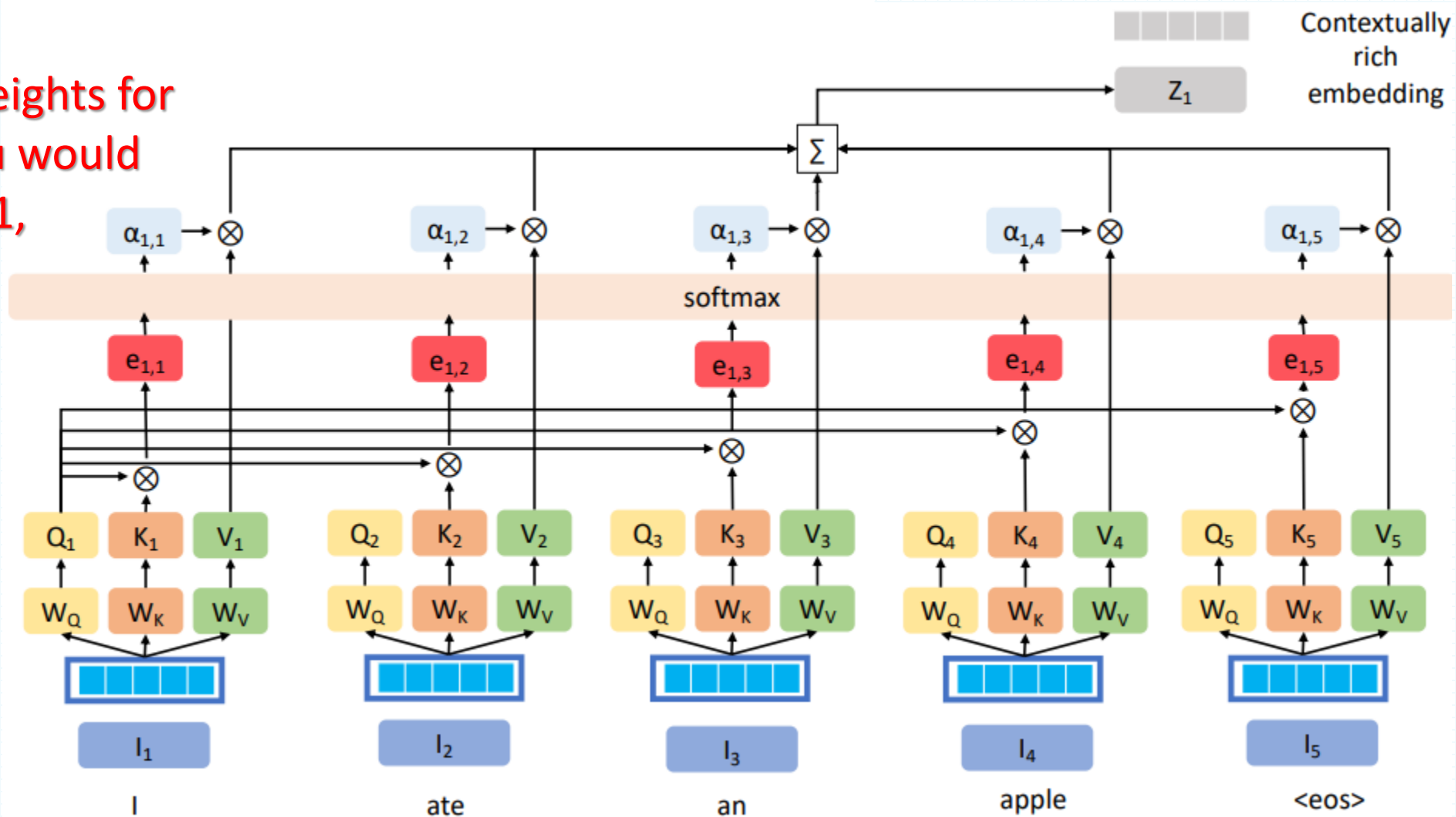
Attention

- More formal!



Attention

To calculate attention weights for input I_1 , you would use query q_1 , and all keys

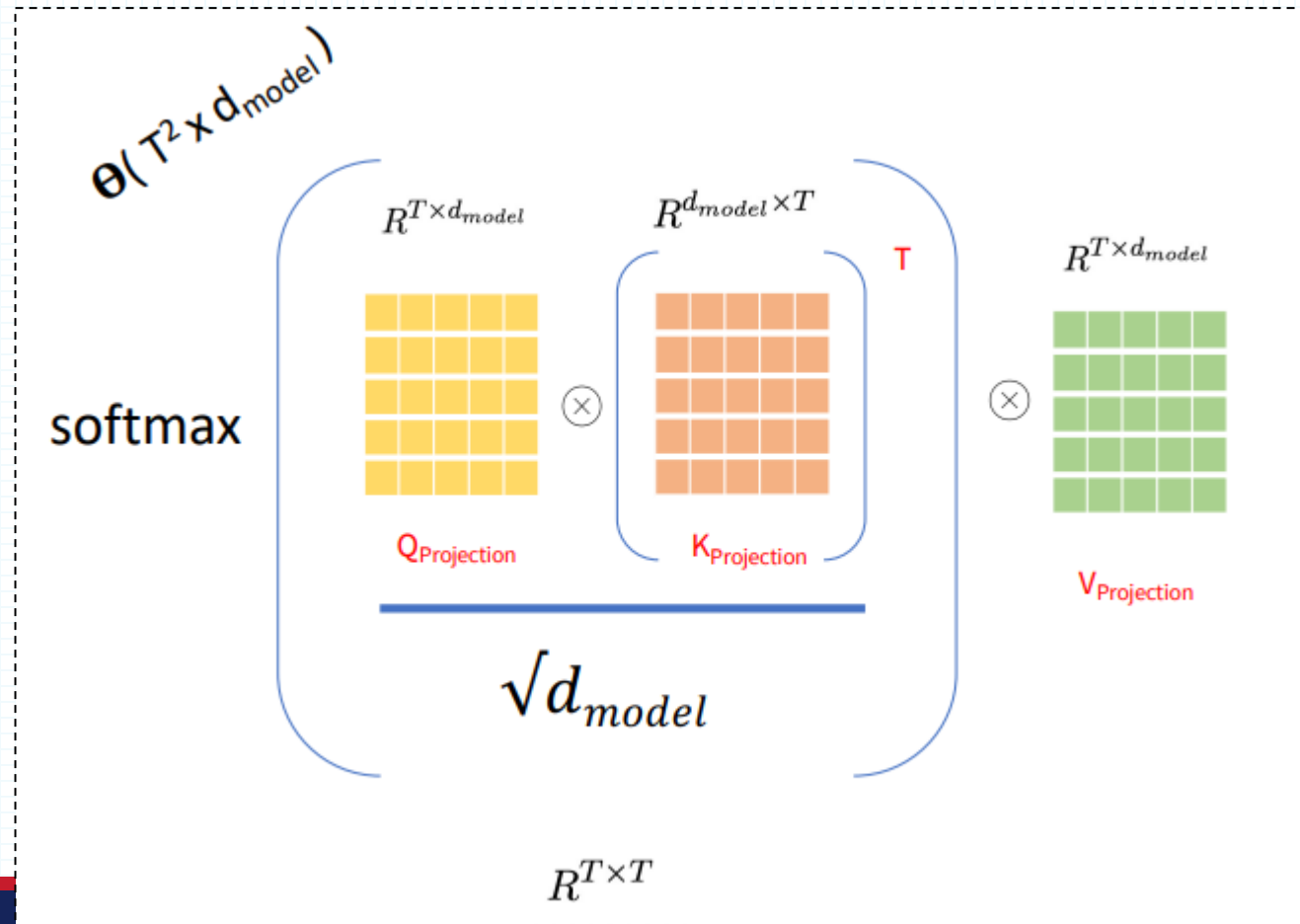


Of I_1

We do the same for I_2 , I_3 , etc

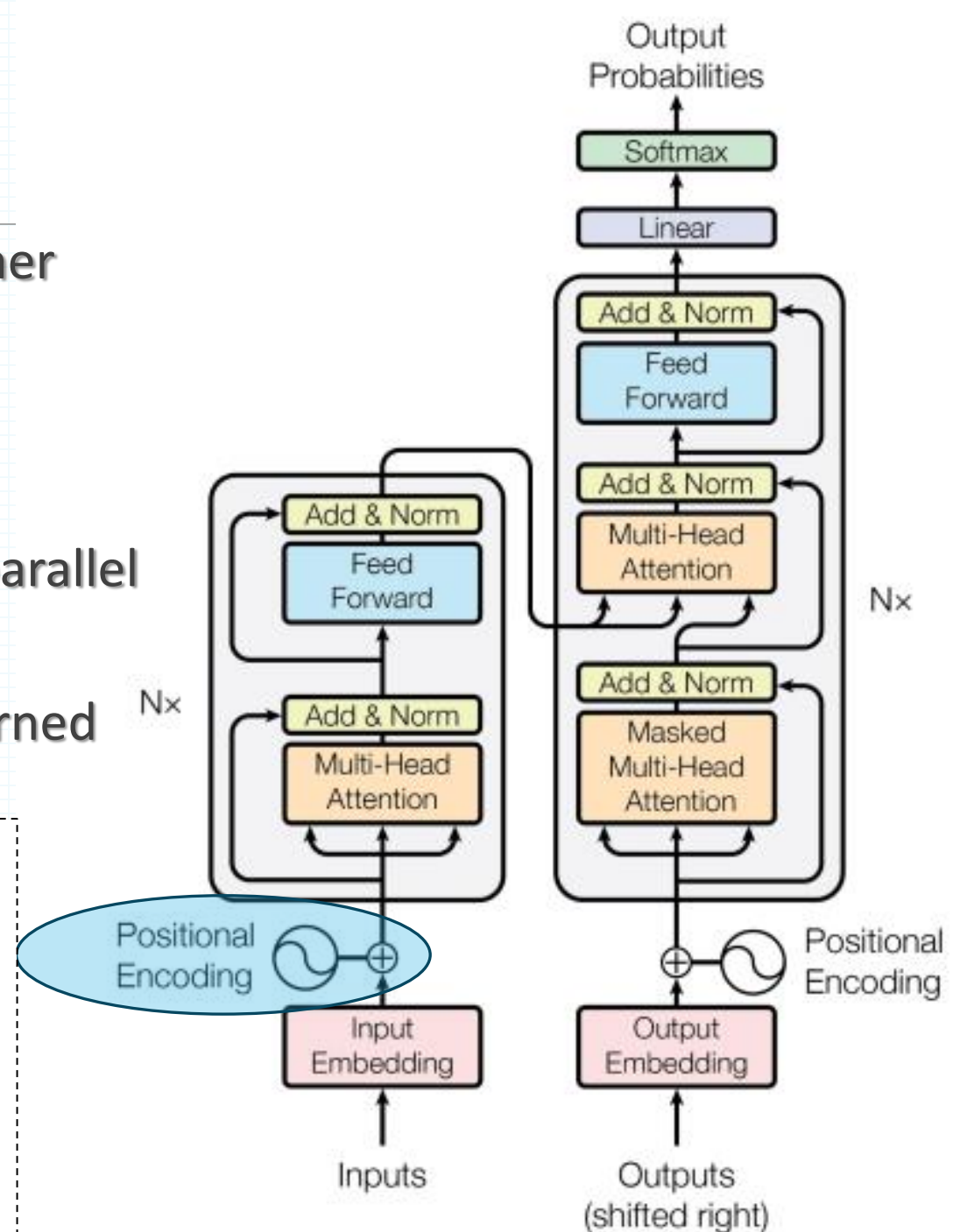
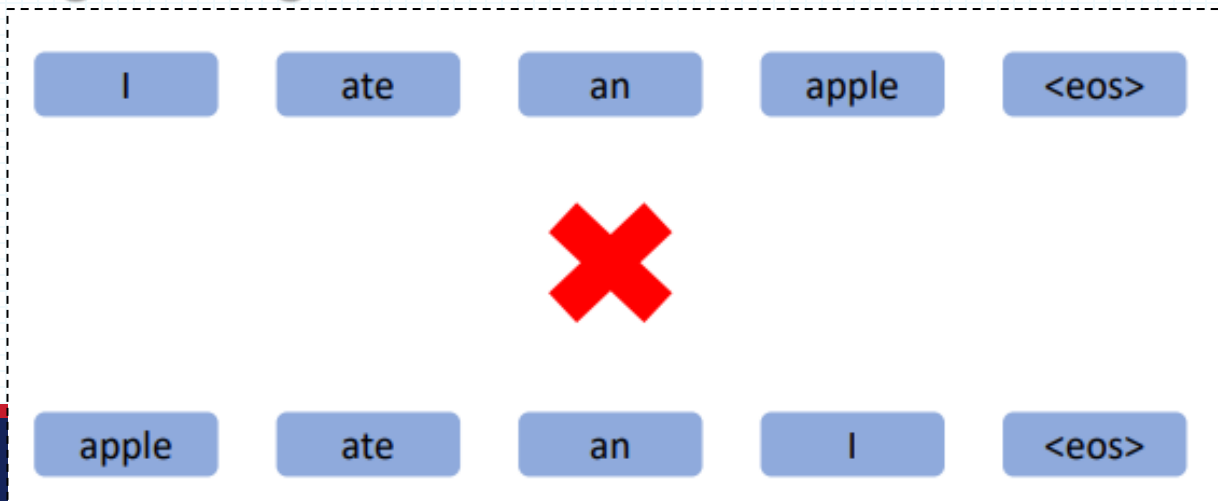
Self Attention

Query Inputs = Key Inputs = Value Inputs



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.



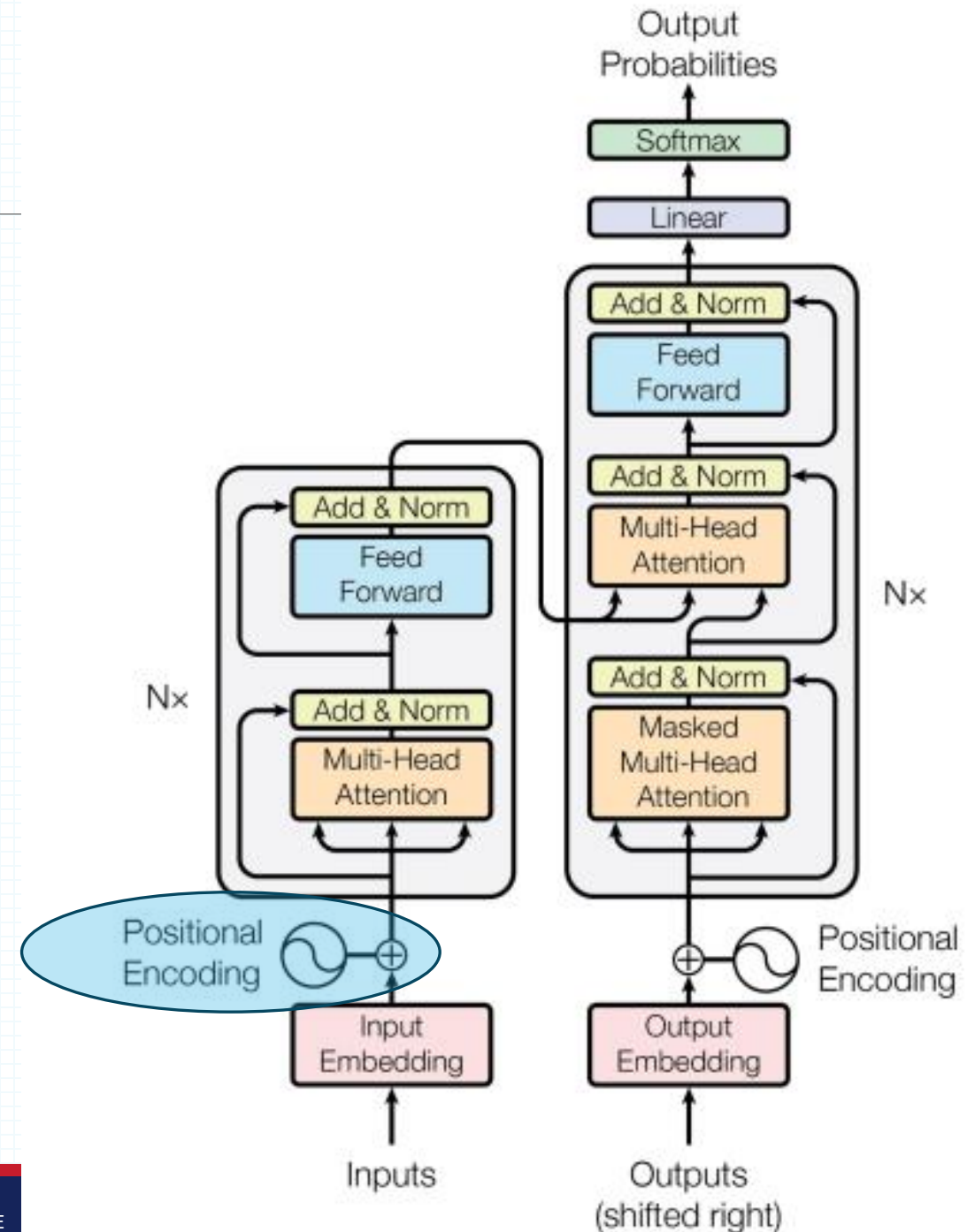
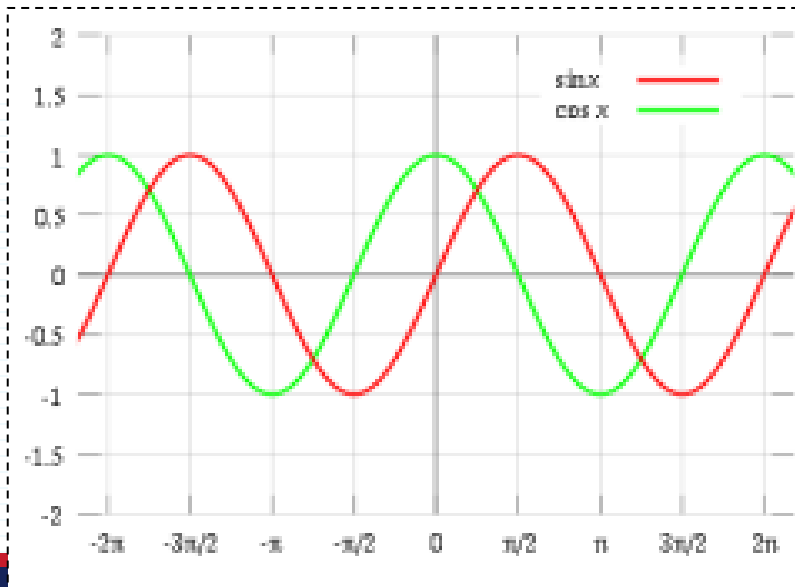
Positional Encoding

pos -> idx of the token in input sentence

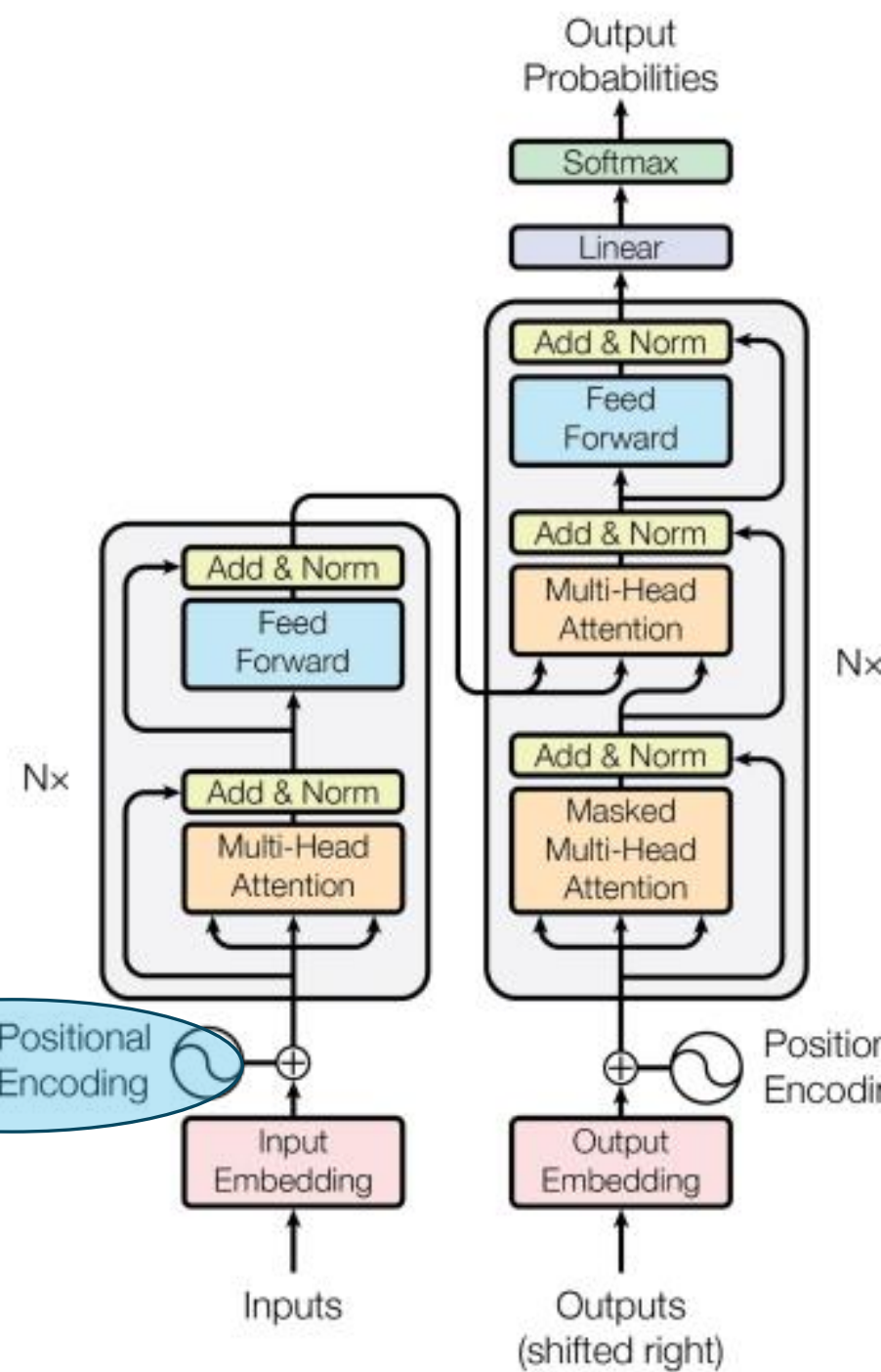
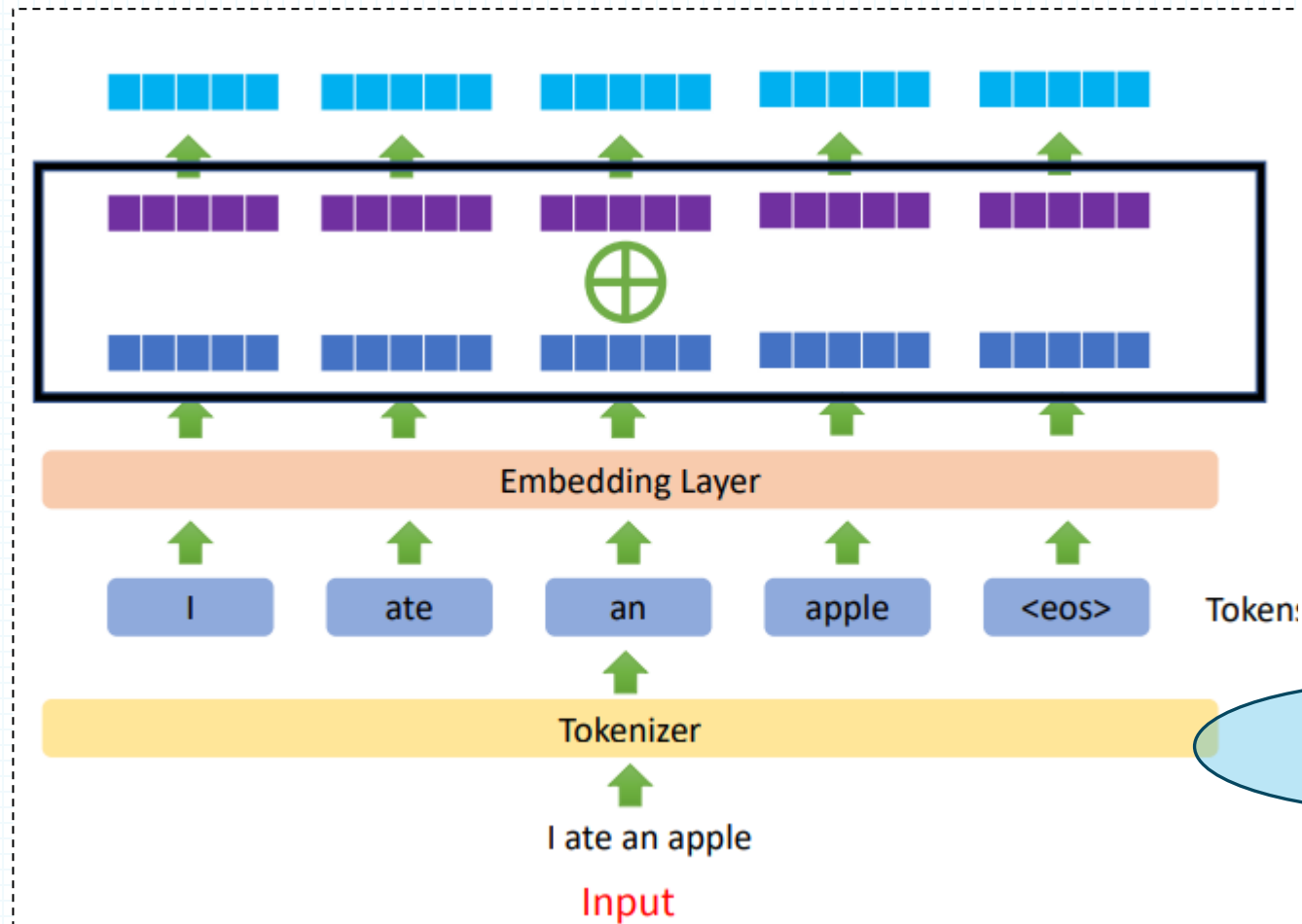
i -> i^{th} dimension out of d

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

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



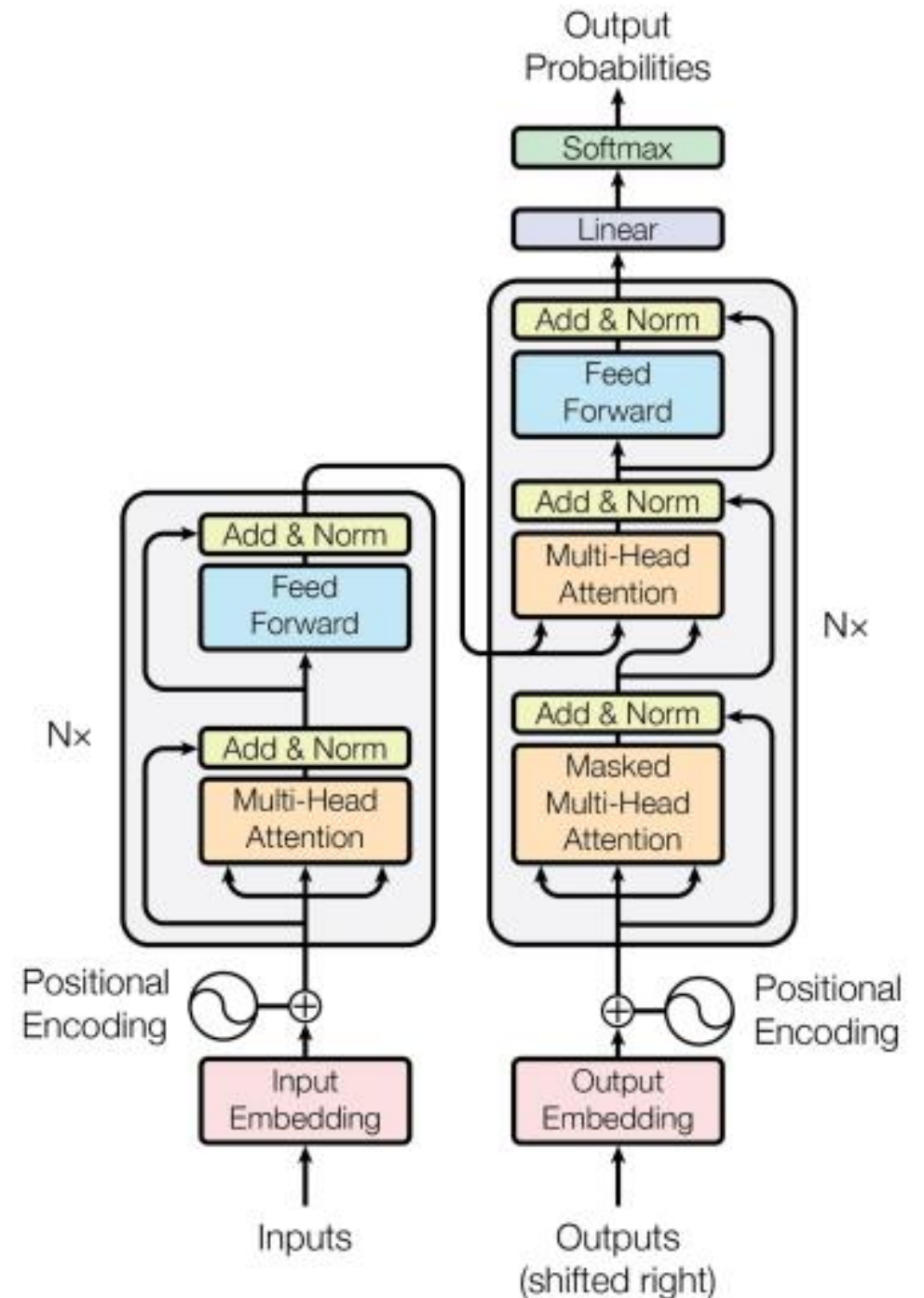
Position Embedding



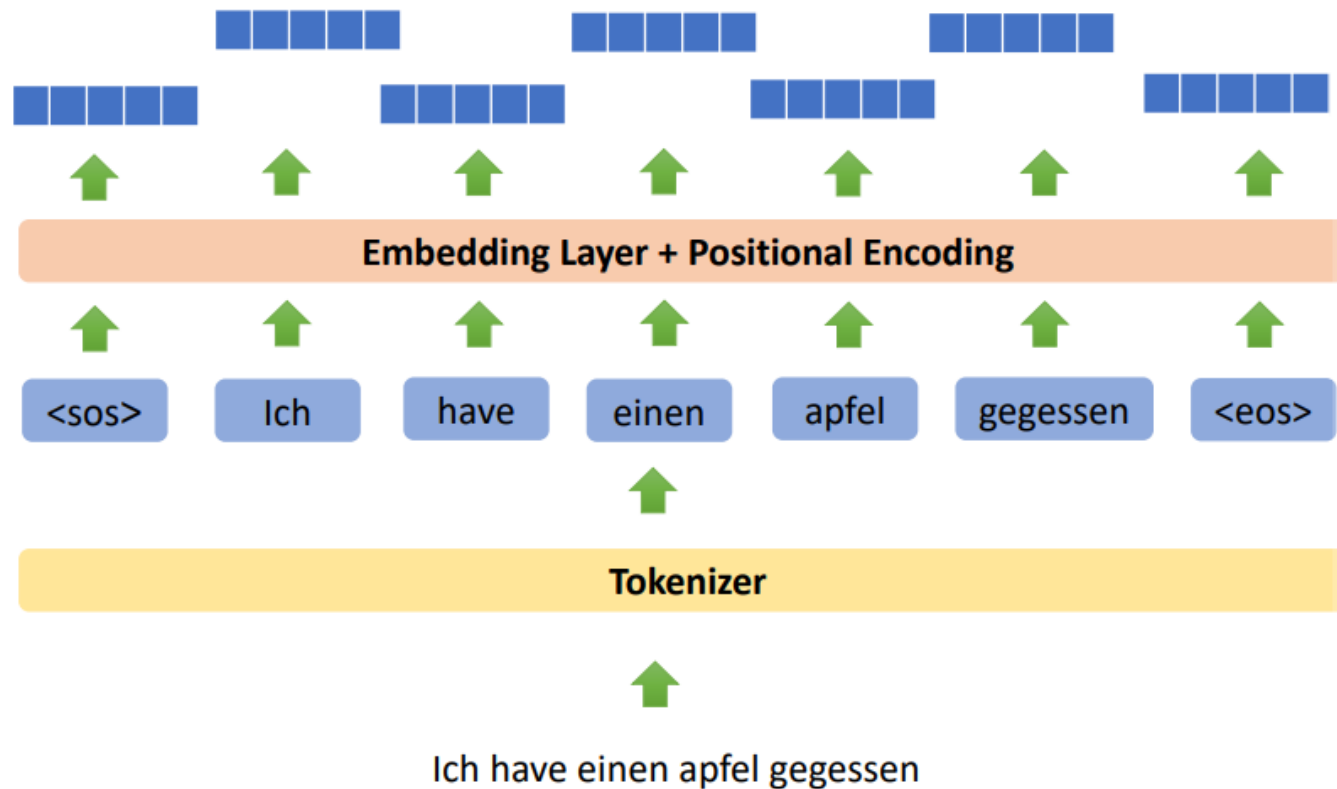
Decoder Part

Transformers

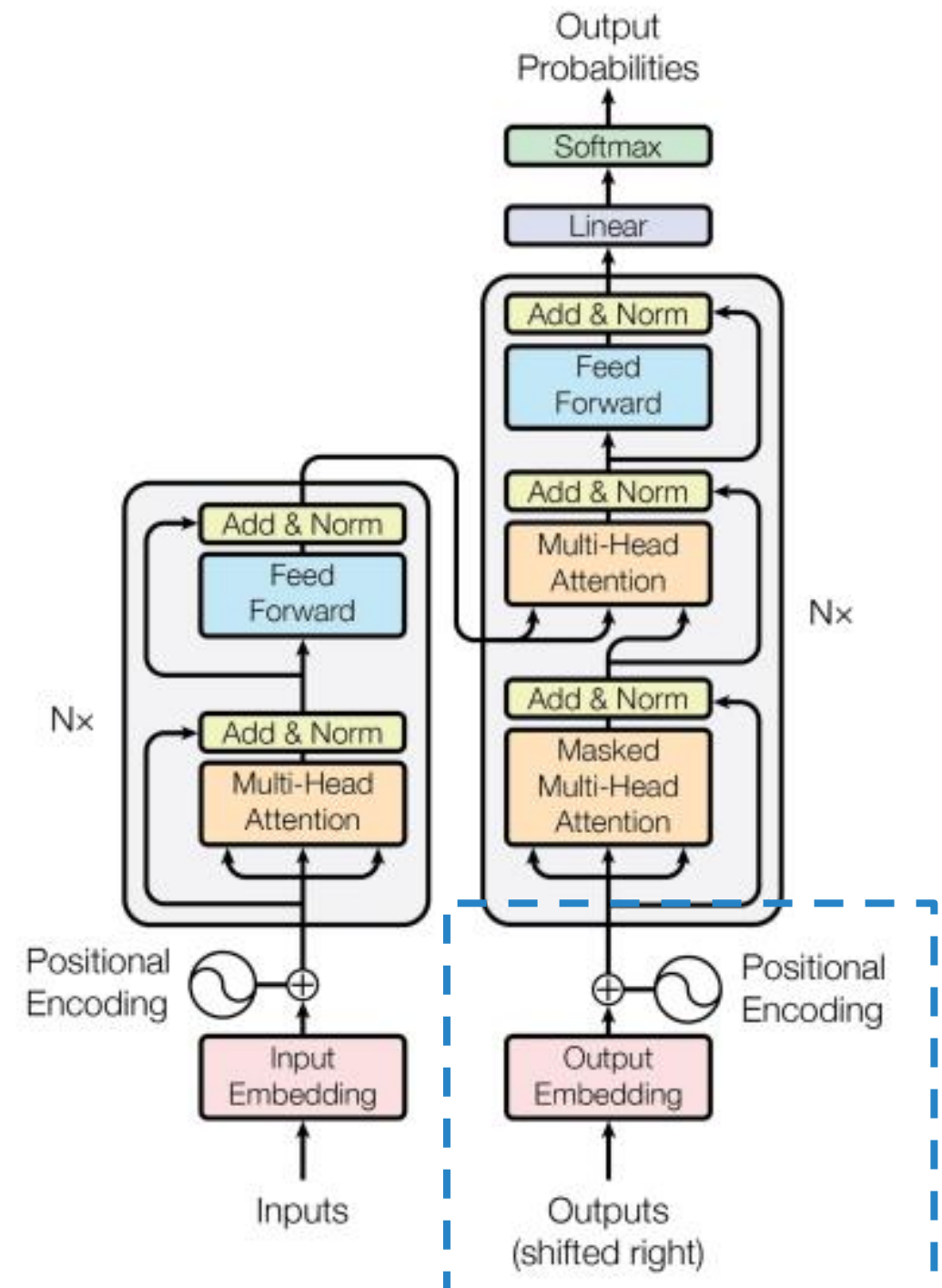
- Transformers are a type of neural network architecture that transforms or changes an input sequence into an output sequence.
- They do this by learning context and tracking relationships between sequence components.
- And break the problem into two parts:
 - An encoder (e.g., Bert)
 - A decoder (e.g.,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



Output Embedding



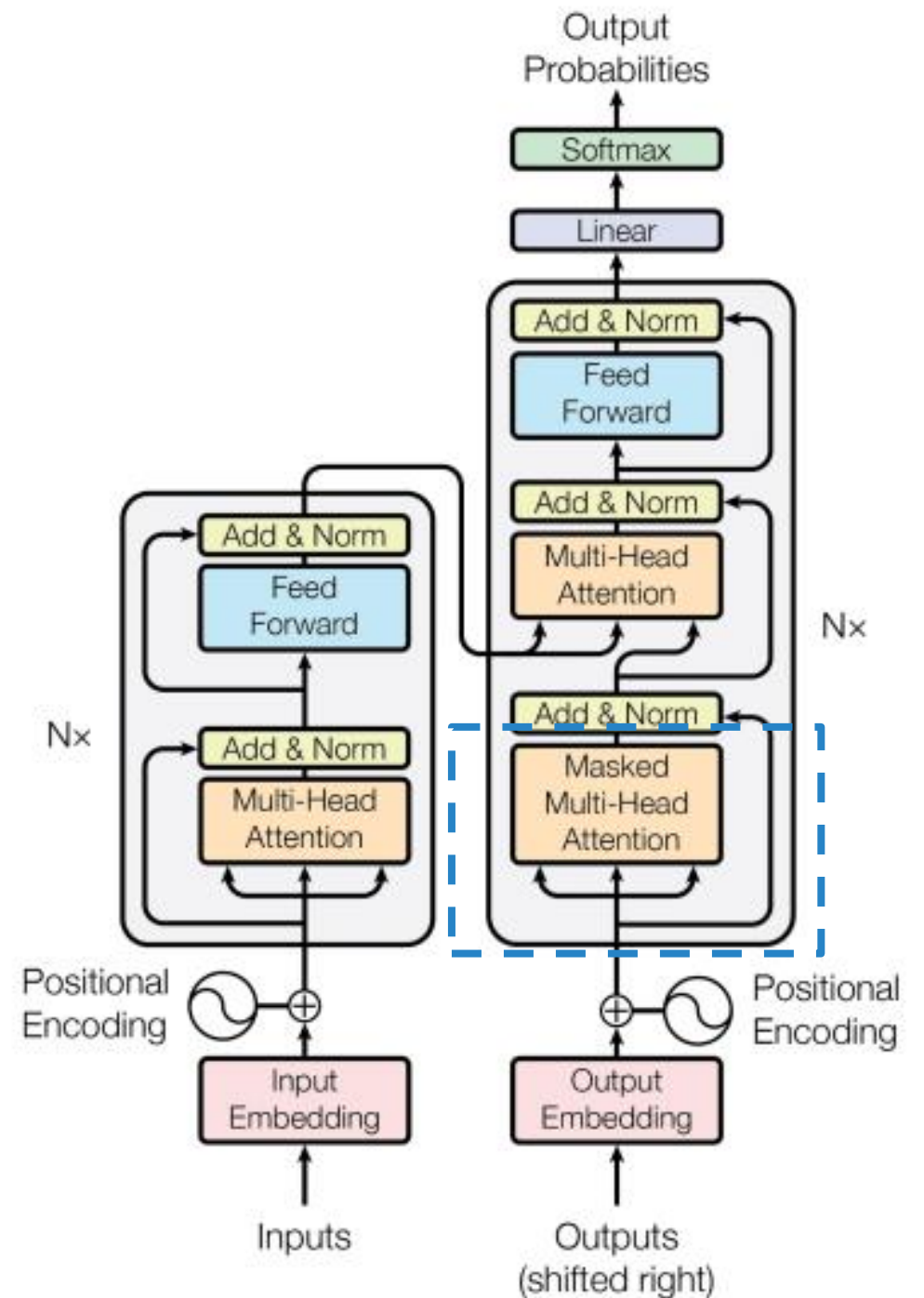
Generate Target Emebeddings



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>

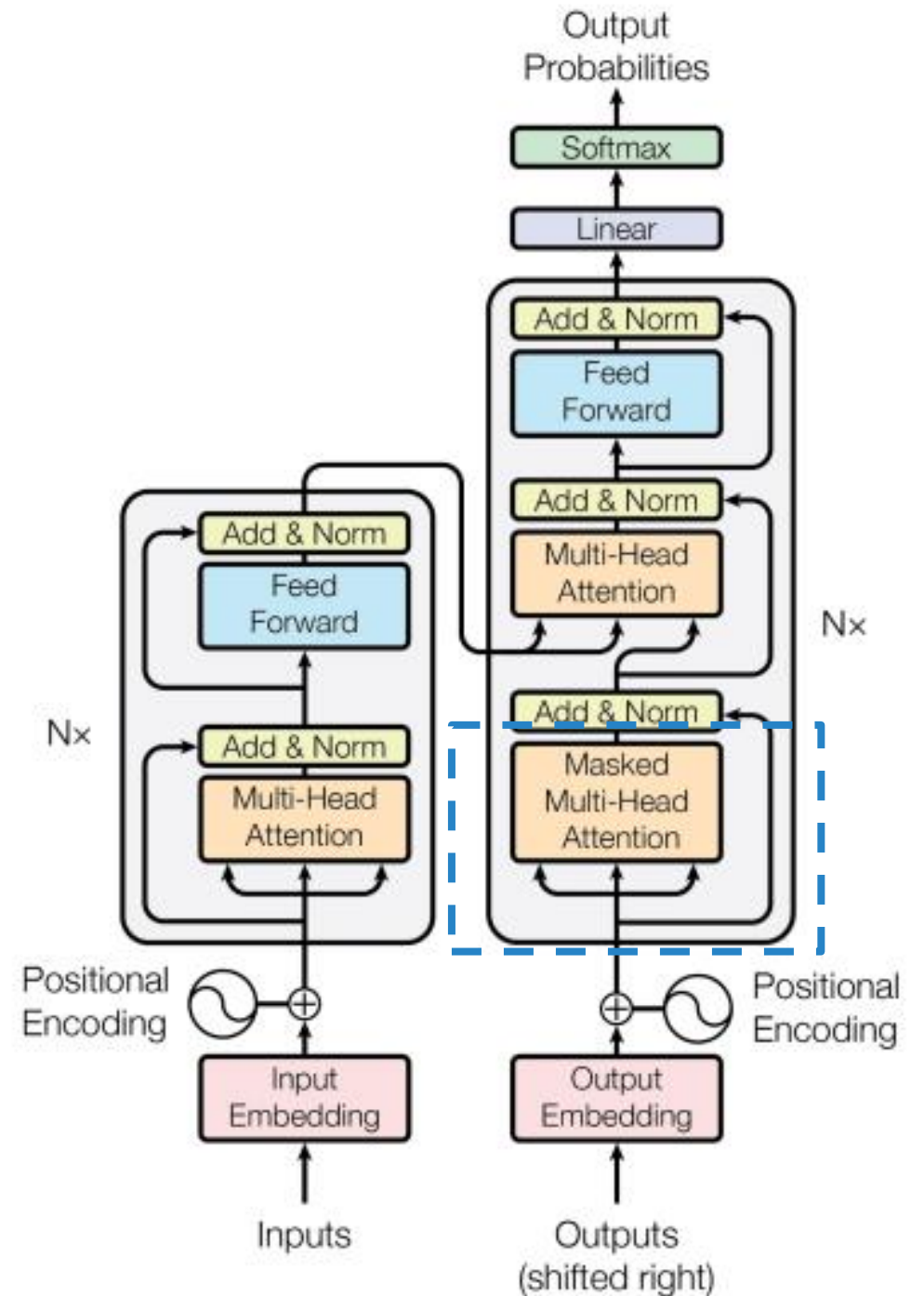
Mask the available attention values ?



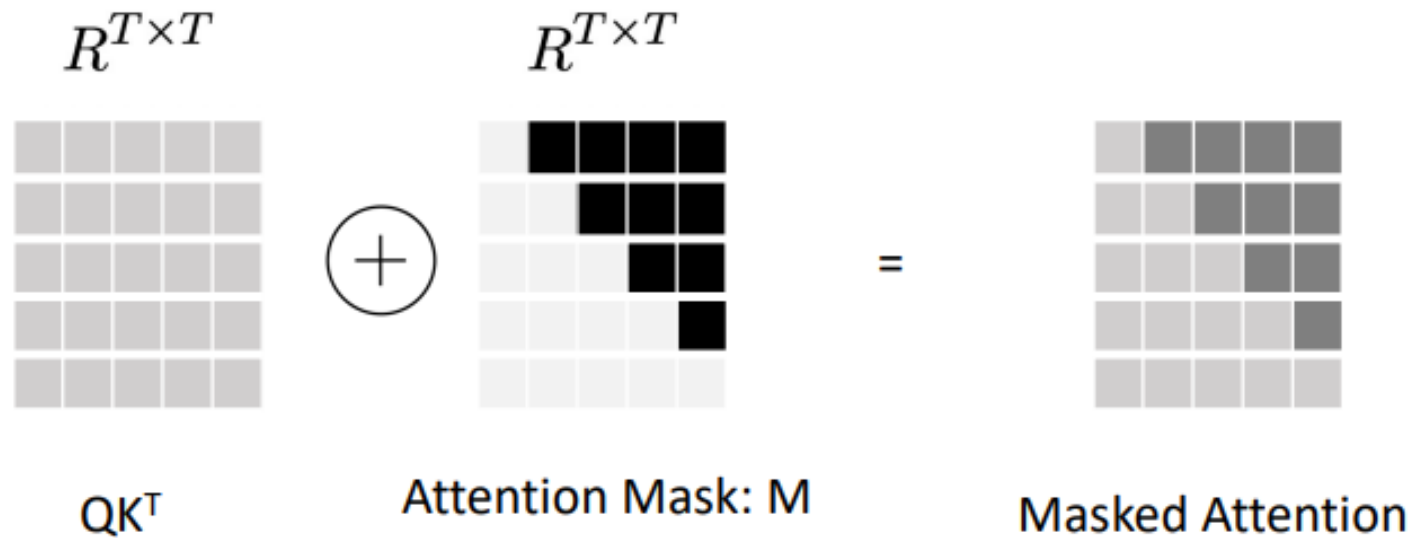
Masked Attention

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

Softmax -> - ∞ -> 0

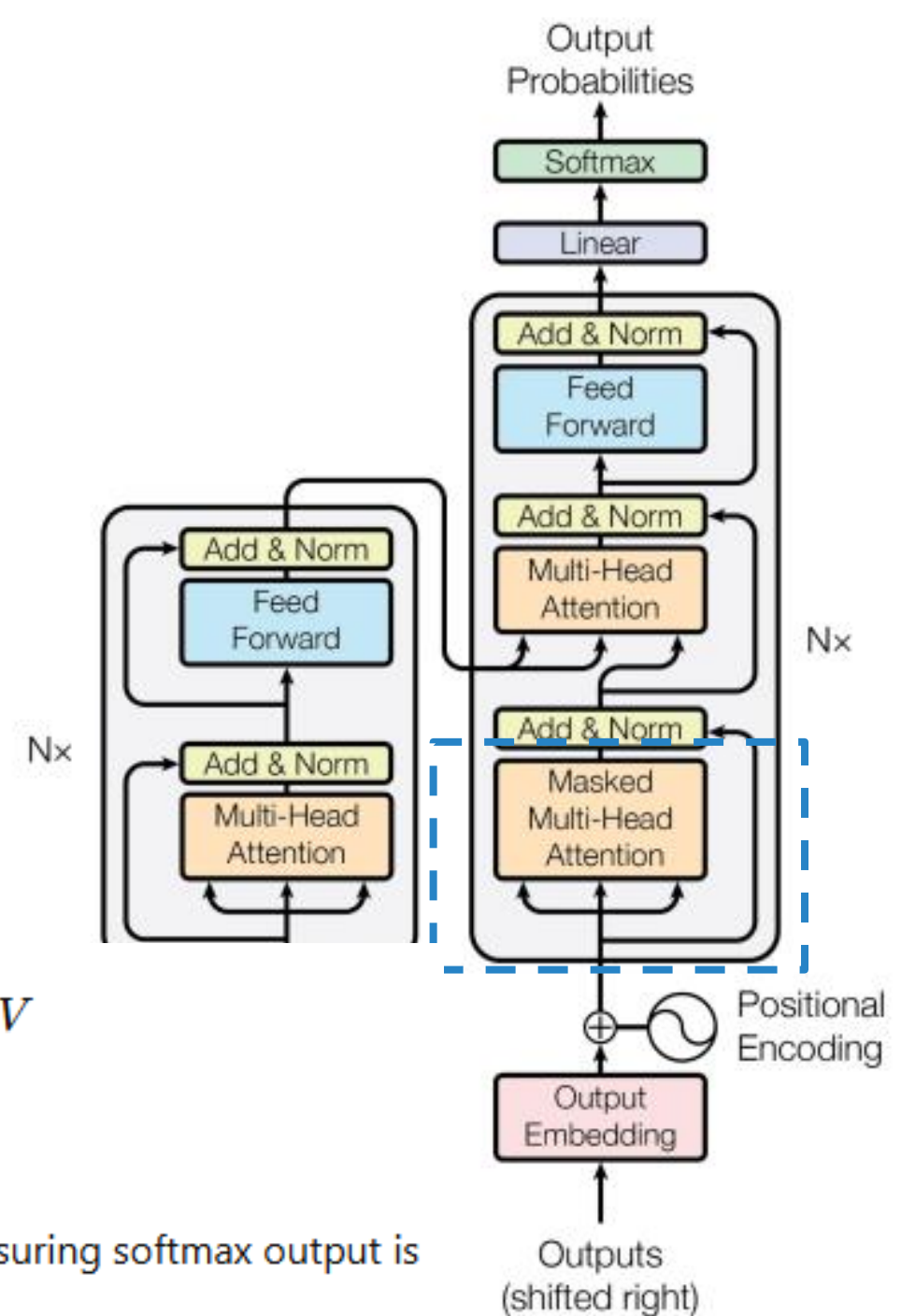


Masked Attention

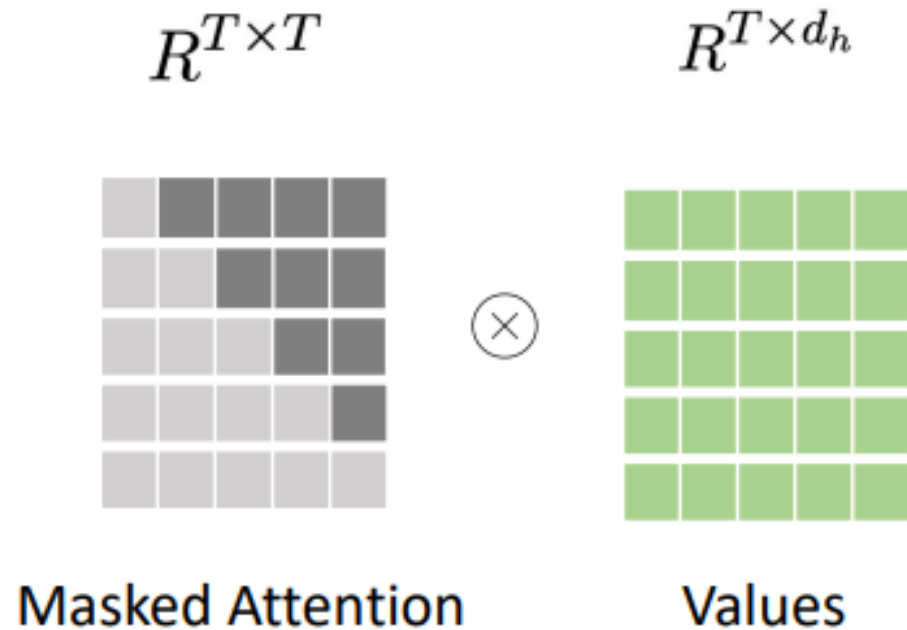


$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} + M \right) V$$

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

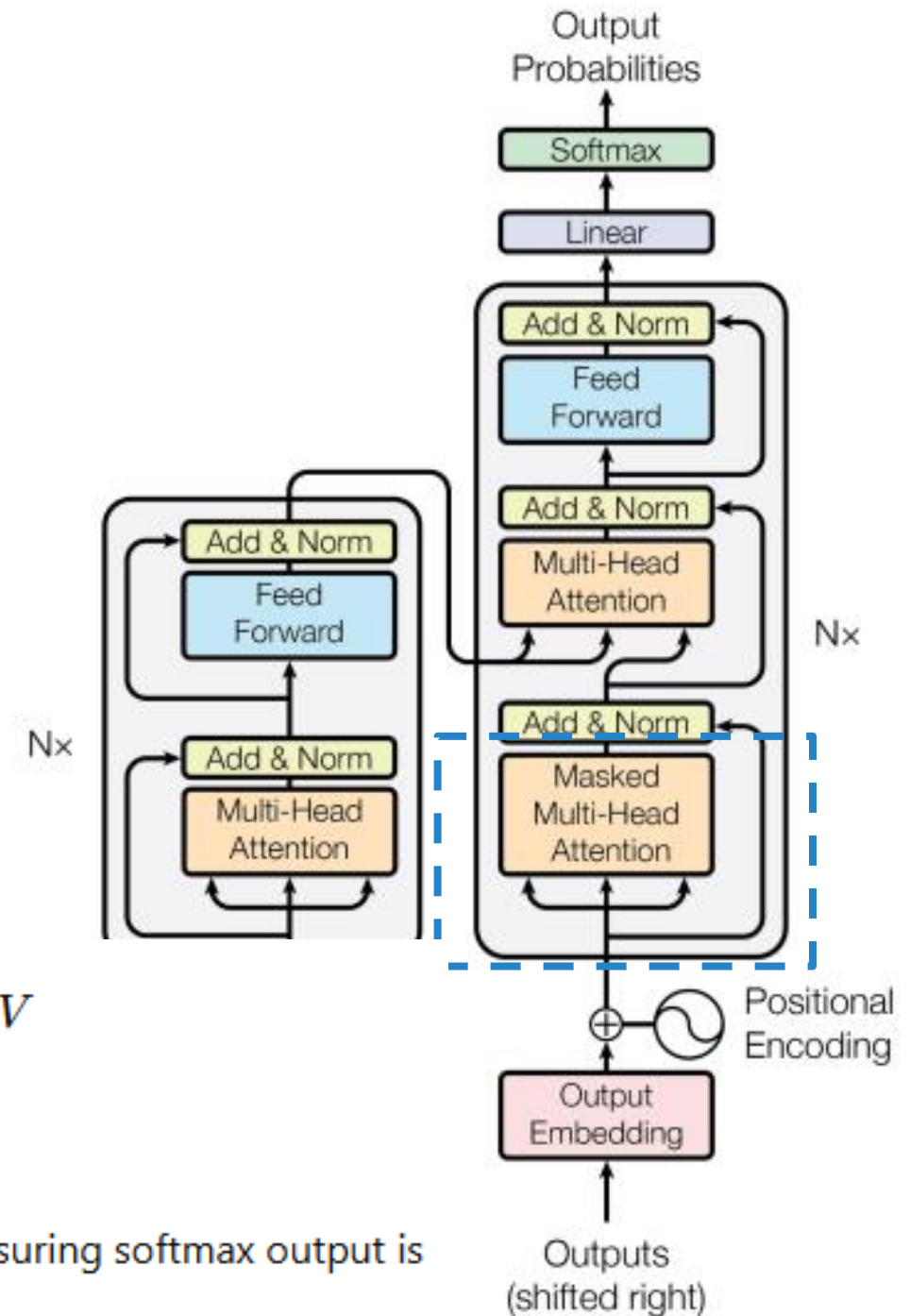


Masked Attention

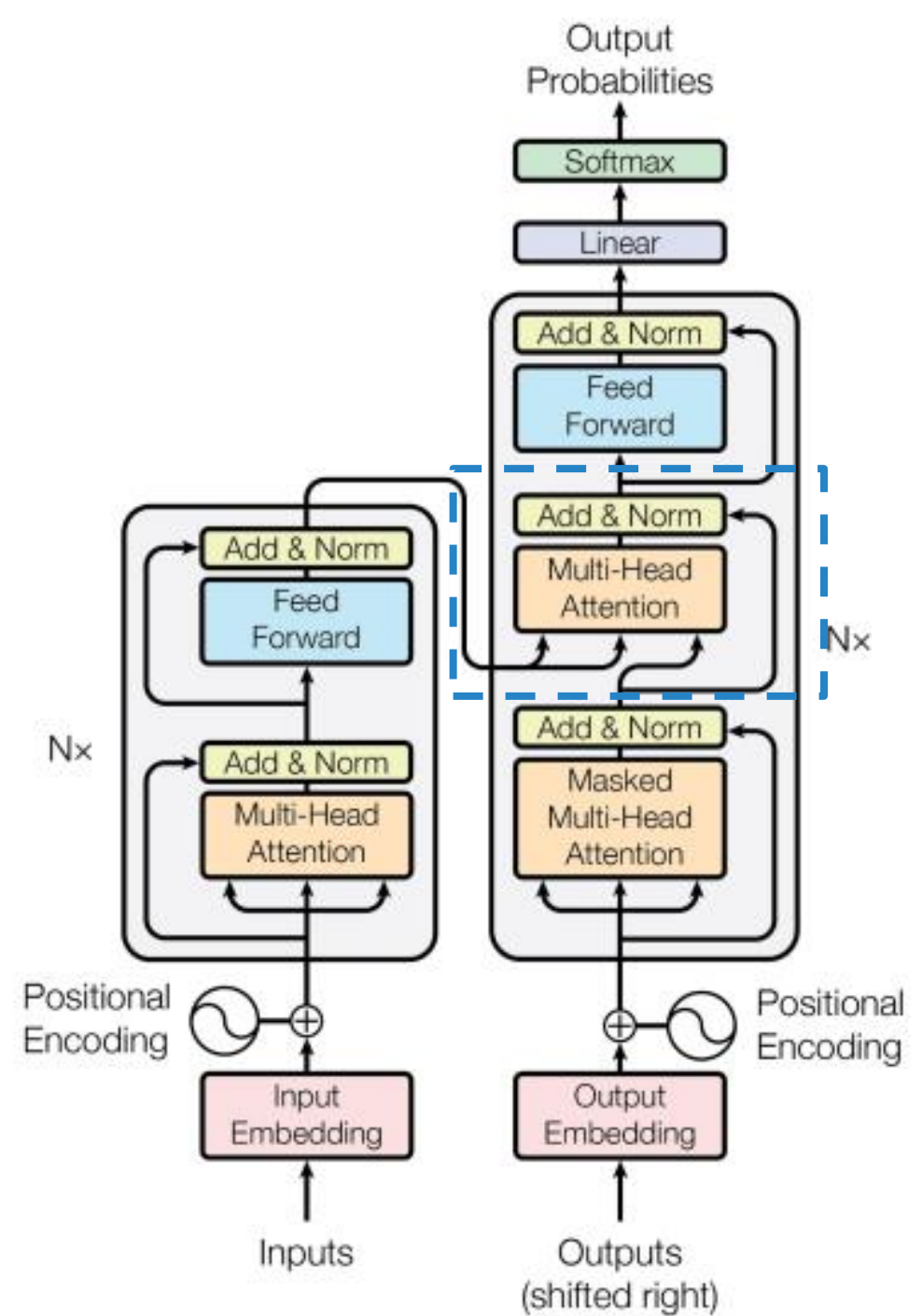
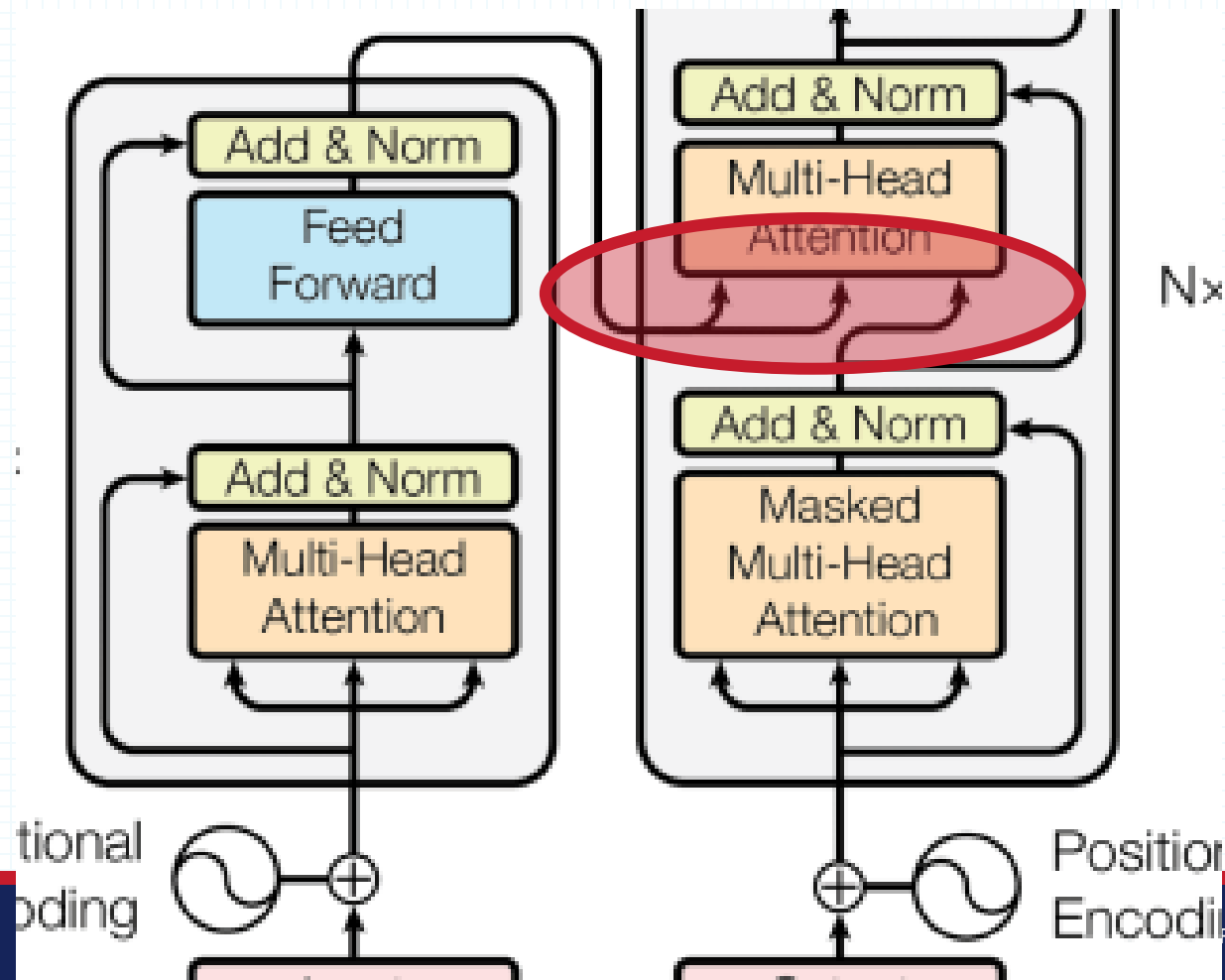


$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} + M \right) V$$

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

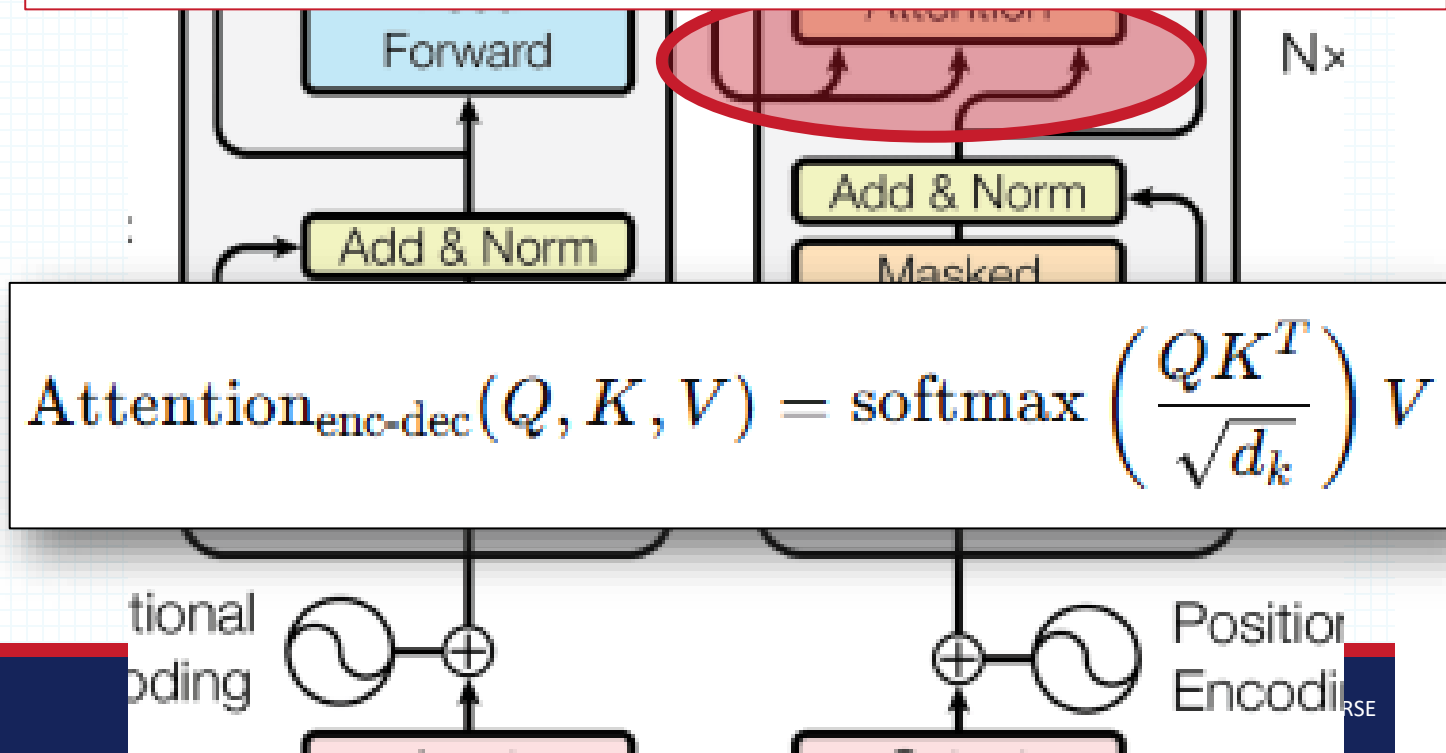


Encoder-Decoder Attention (Cross-Attention)

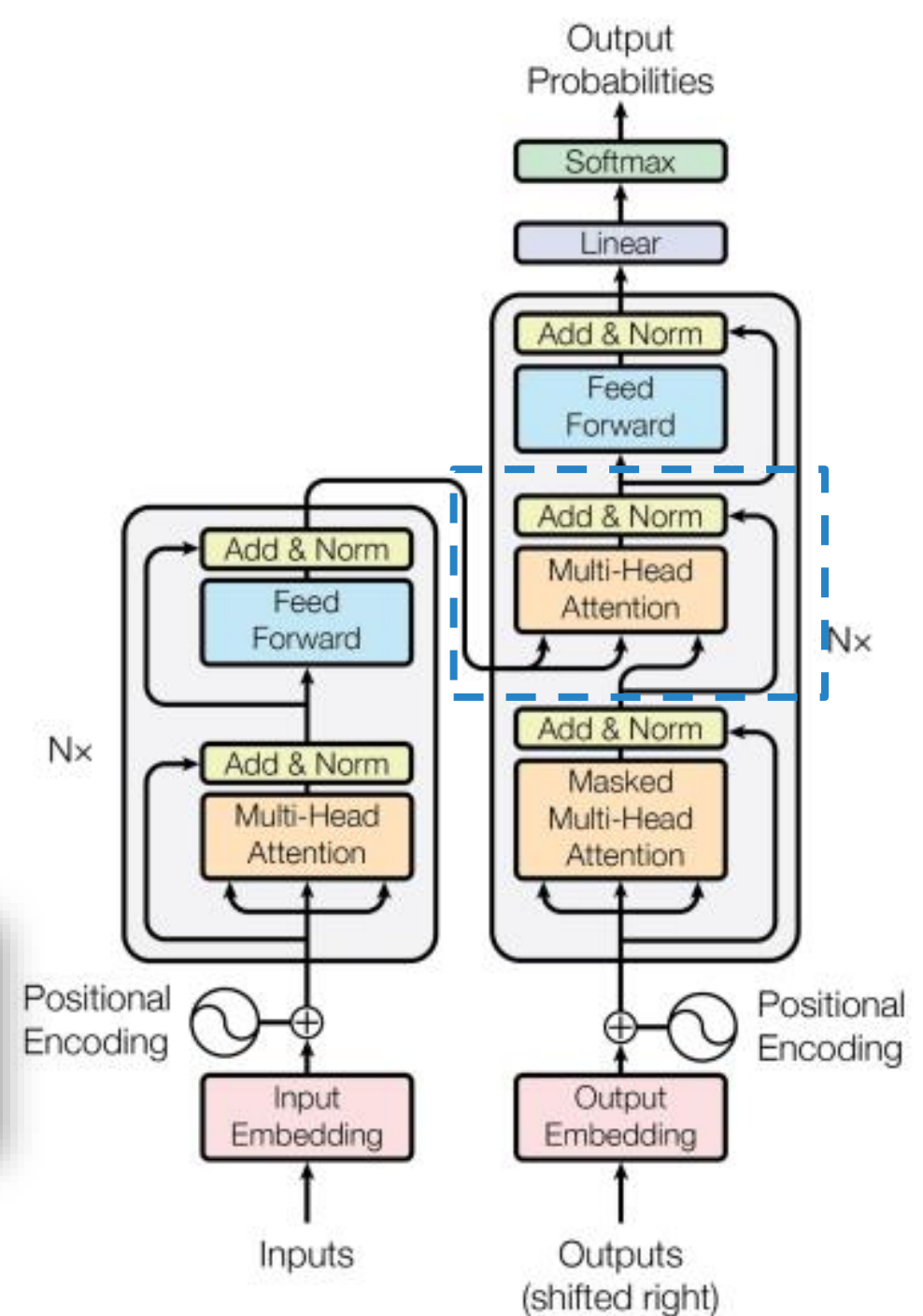


Encoder-Decoder Attention (Cross-Attention)

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



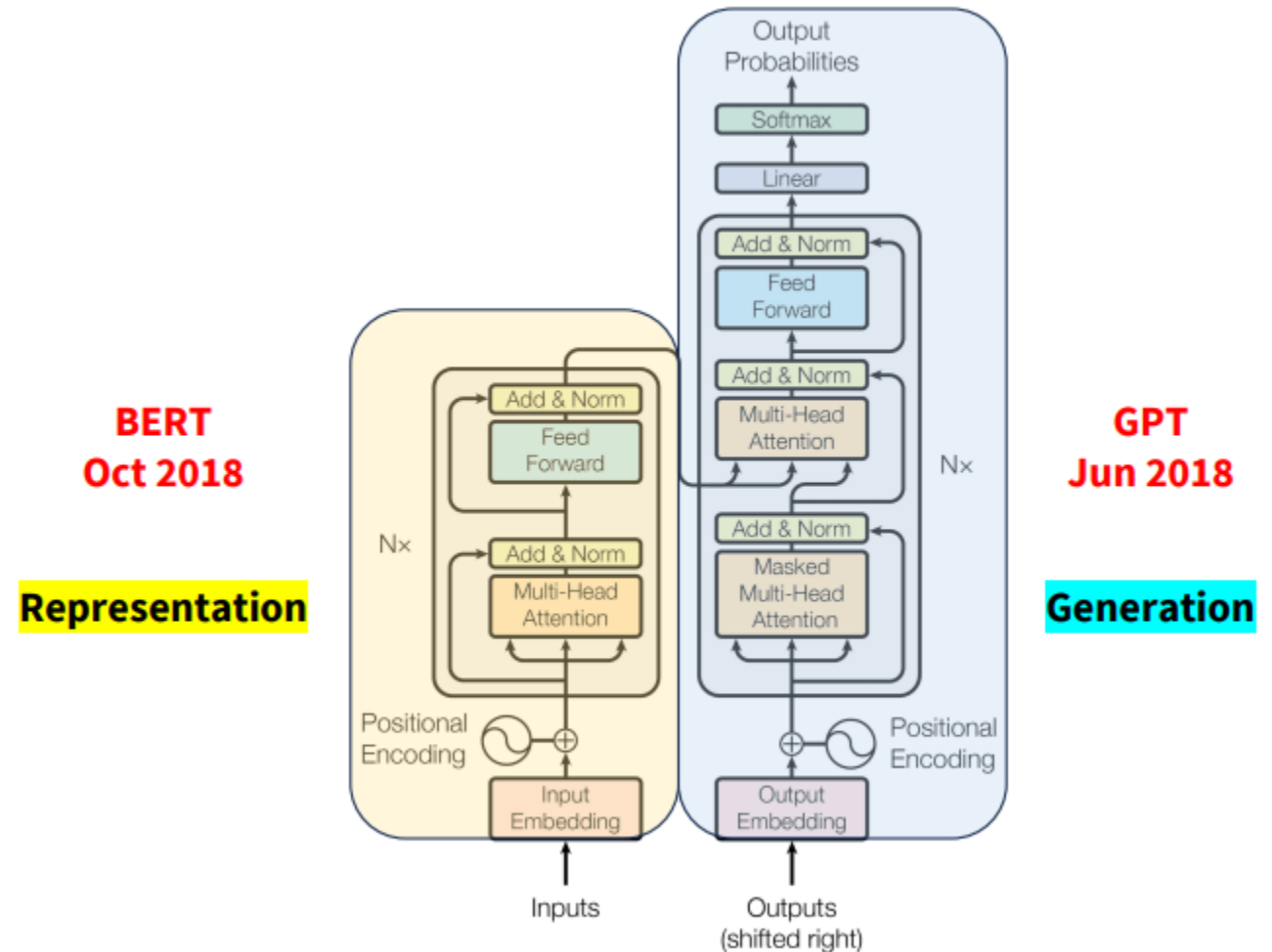
$$\text{Attention}_{\text{enc-dec}}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$



Post-Transformer Evolution and Milestones

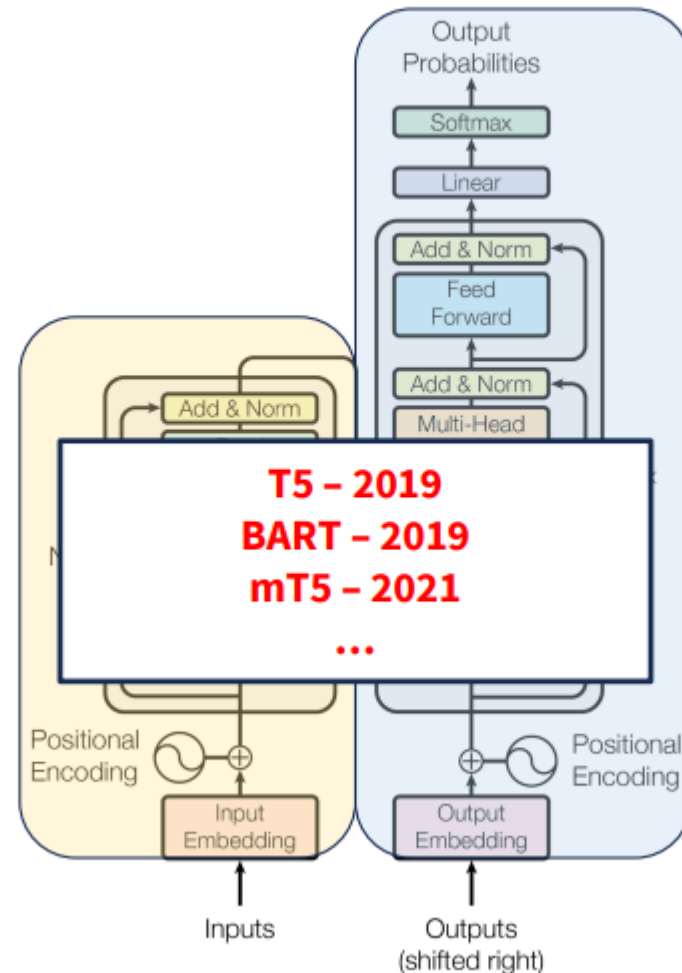
Transformers

- 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)



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