Deep Learning

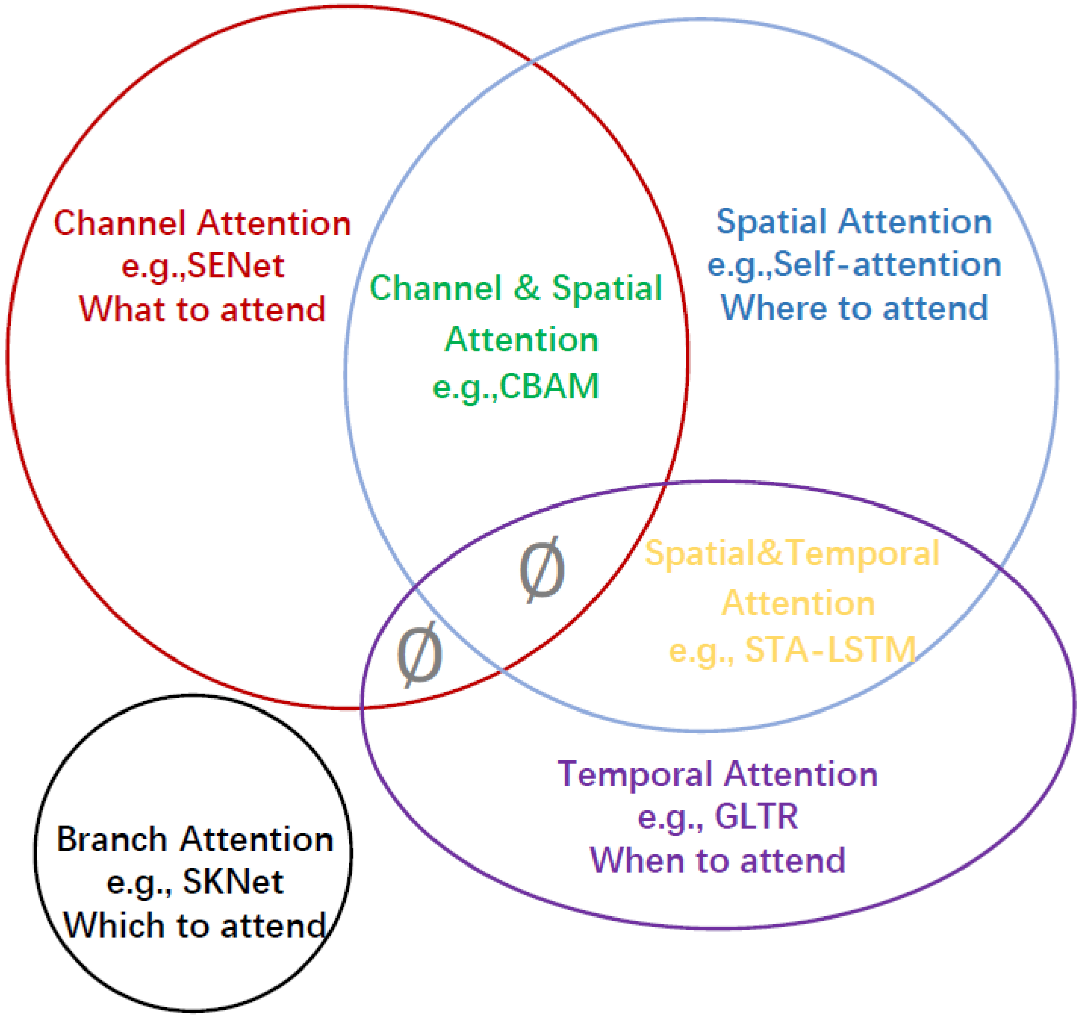
Angshuman Paul

Assistant Professor

Department of Computer Science & Engineering

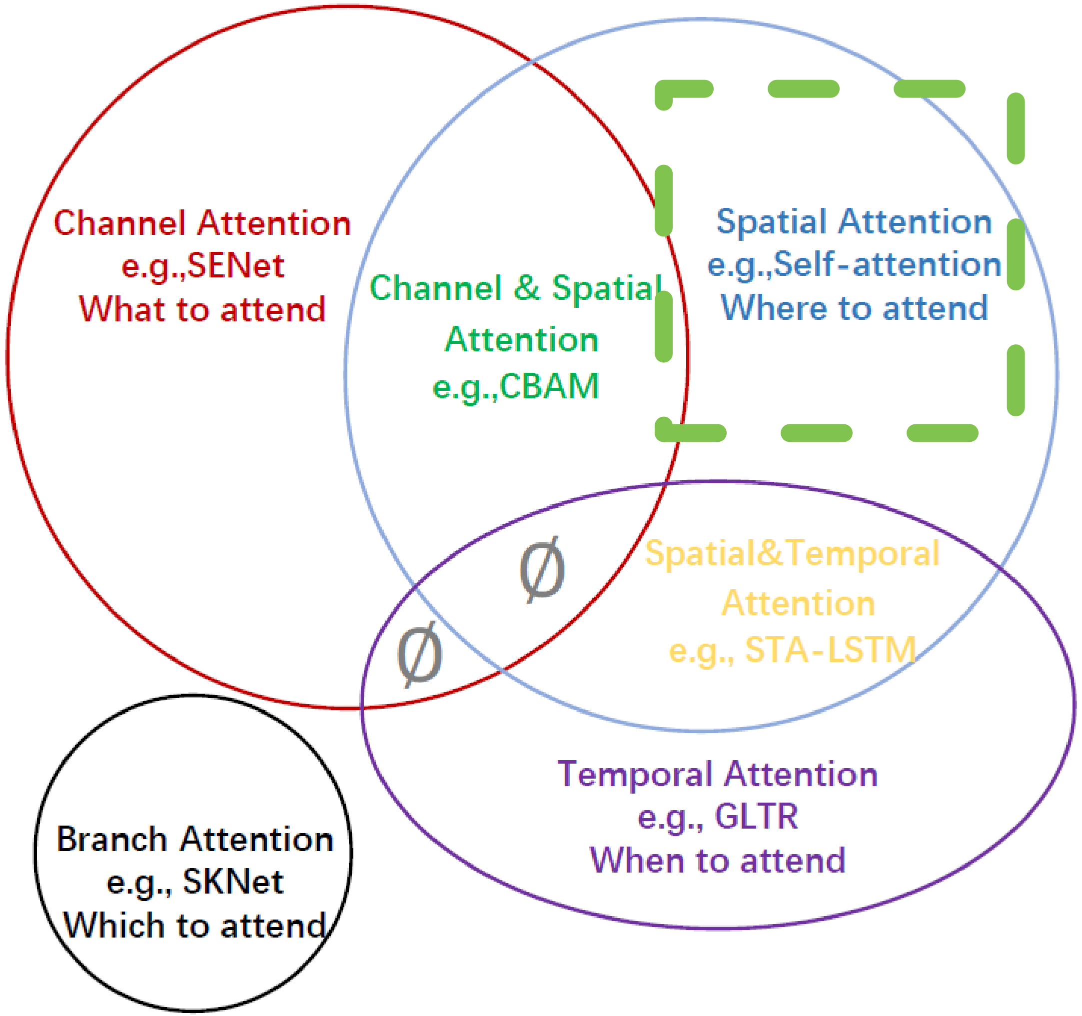
Transformers

Attention in Research Papers



Guo *et al.*, “Attention mechanisms in computer vision: A survey”. Computational Visual Media. 2022

Attention in Research Papers



Guo *et al.*, "Attention mechanisms in computer vision: A survey". Computational Visual Media. 2022

Attention in Language Modelling

- 2017: Attention is All You Need (Vaswani *et al.*)
- Introduction of Transformer

Attention Mechanism

- Mimics the process of retrieval of a value v_i based on key k_i from a database for a query q

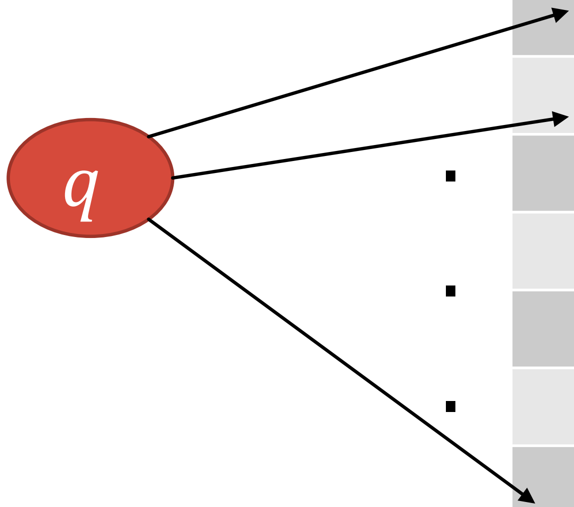
Attention Mechanism

- Mimics the process of retrieval of a value v_i based on key k_i from a database for a query q

Key	Value
k_1	v_1
k_2	v_2
.	.
.	.
.	.
.	.
k_n	v_n

Attention Mechanism

- Mimics the process of retrieval of a value v_i based on key k_i from a database for a query q



Key	Value
k_1	v_1
k_2	v_2
.	.
.	.
.	.
.	.
k_n	v_n

Attention Mechanism

- Mimics the process of retrieval of a value v_i based on key k_i from a database for a query q

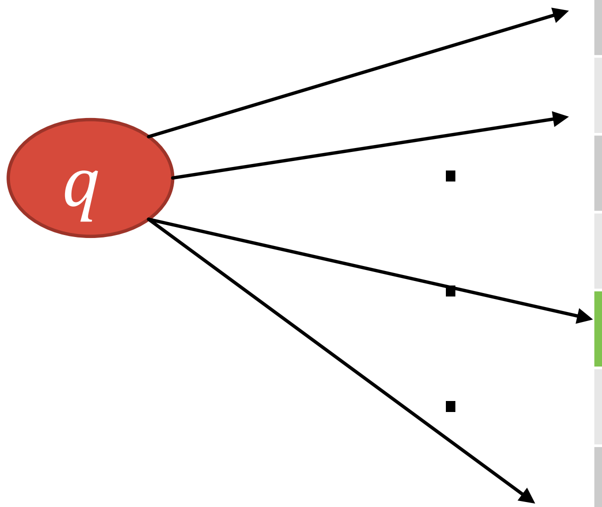
q

Key	Value
k_1	v_1
k_2	v_2
.	.
.	.
.	.
.	.
k_n	v_n

Compute similarity between query and keys

Attention Mechanism

- Mimics the process of retrieval of a value v_i based on key k_i from a database for a query q

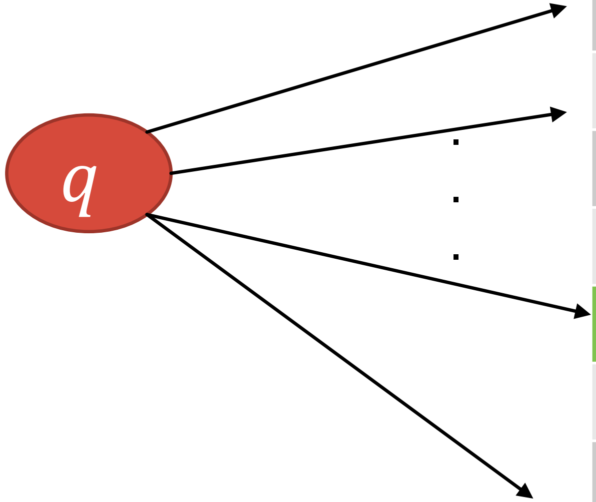


Similarity	Key	Value
0.01	k_1	v_1
0.03	k_2	v_2
.	.	.
.	.	.
0.87	$\boldsymbol{k_m}$	$\boldsymbol{v_m}$
.	.	.
k_n	k_n	v_n

Compute similarity between query and keys

Attention Mechanism

- Mimics the process of retrieval of a value v_i based on key k_i from a database for a query q

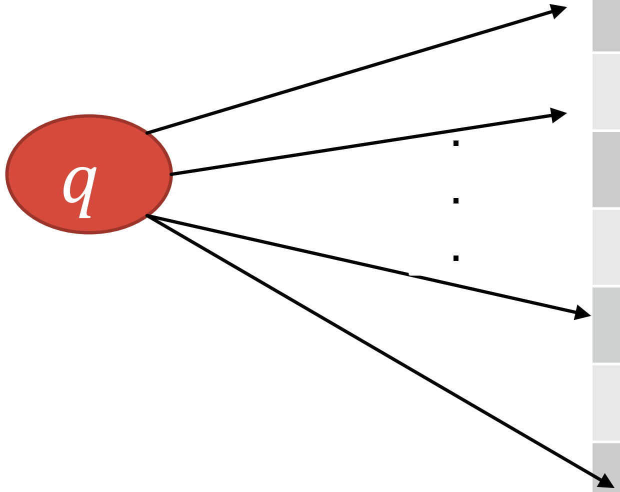


Similarity	Key	Value
0.01	k_1	v_1
0.03	k_2	v_2
.	.	.
.	.	.
0.87	$\boldsymbol{k_m}$	$\boldsymbol{v_m}$
.	.	.
k_n	k_n	v_n

v_m is the output of the query

Attention Mechanism

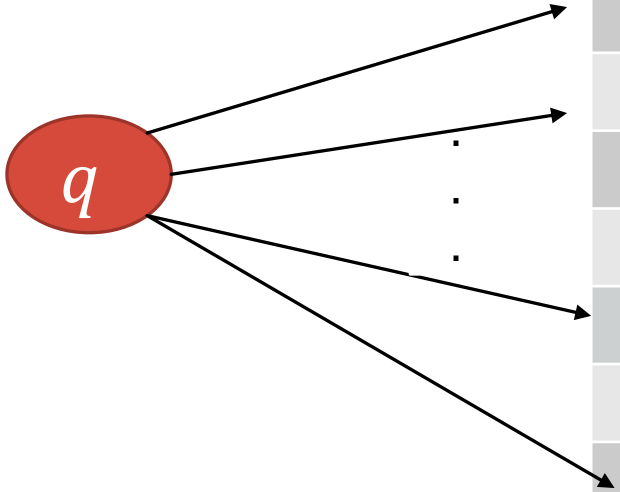
- What else can we do to get the output?



Similarity	Key	Value
s_1	k_1	v_1
s_2	k_2	v_2
.	.	.
.	.	.
s_m	k_m	v_m
.	.	.
s_n	k_n	v_n

Attention Mechanism

- Instead of a single output, we may take a weighted combination of the outputs

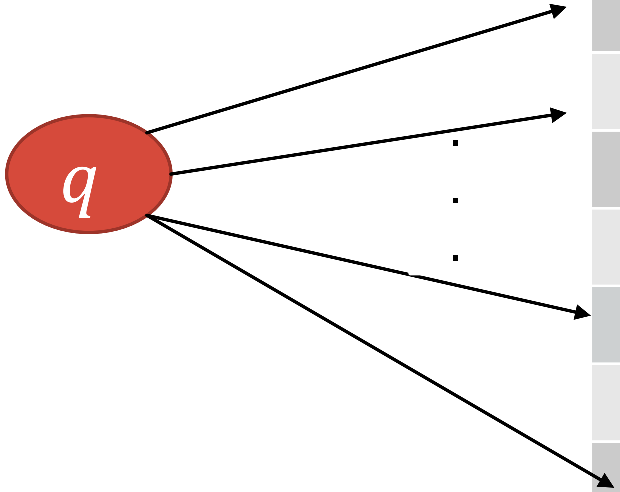


Similarity	Key	Value
s_1	k_1	v_1
s_2	k_2	v_2
.	.	.
.	.	.
s_m	k_m	v_m
.	.	.
s_n	k_n	v_n

$$Output = \sum_i softmax(s_i)v_i$$

Attention Mechanism

- Instead of a single output, we may take a weighted combination of the outputs



Similarity	Key	Value
s_1	k_1	v_1
s_2	k_2	v_2
.	.	.
.	.	.
s_m	k_m	v_m
.	.	.
s_n	k_n	v_n

$$Output = \sum_i softmax(s_i)v_i$$

$$= \sum_i softmax(similarity(q, k_i)) \times v_i$$

How to Calculate Similarity?

- Instead of a single output, we may take a weighted combination of the outputs

$similarity(q, k_i) = q^T k_i$ **Dot product**

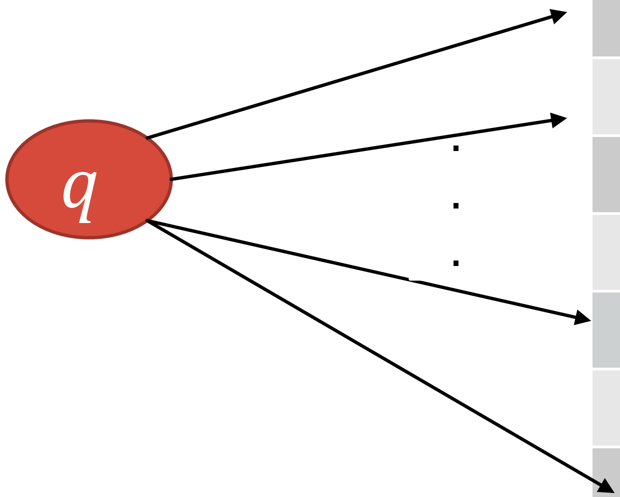
$similarity(q, k_i) = \frac{q^T k_i}{\sqrt{d}}$ **Scaled Dot product**

d is the dimensionality

$similarity(q, k_i) = q^T W k_i$ **General Dot product**

Attention Mechanism

- Attention value can be computed using similarity scores and values



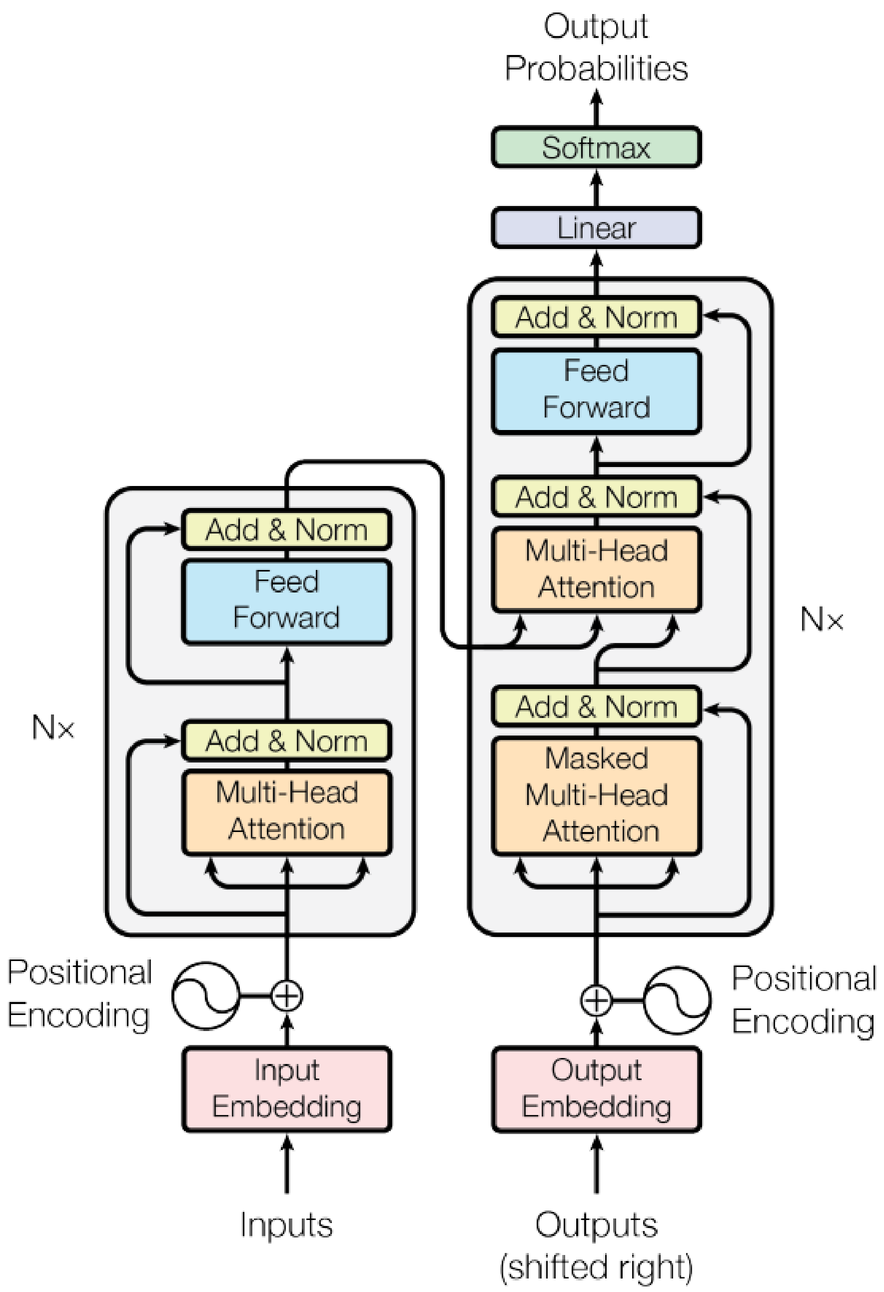
Similarity	Key	Value
s_1	k_1	v_1
s_2	k_2	v_2
.	.	.
.	.	.
s_m	k_m	v_m
.	.	.
s_n	k_n	v_n

$$Attention\ value = \sum_i softmax(s_i)v_i$$

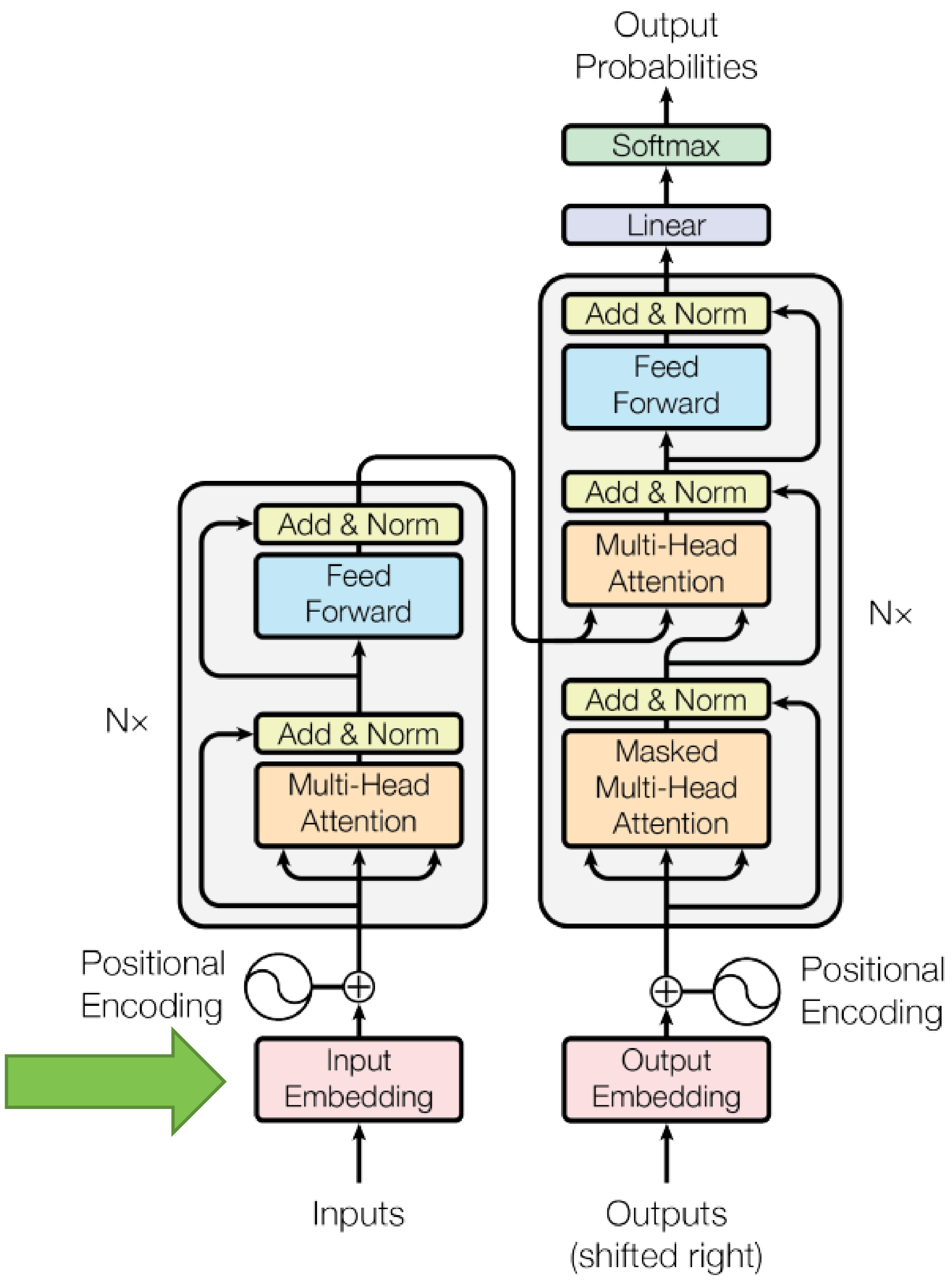
Transformer for NLP

Where there is will, there ....

Can an ML model complete the sentence?



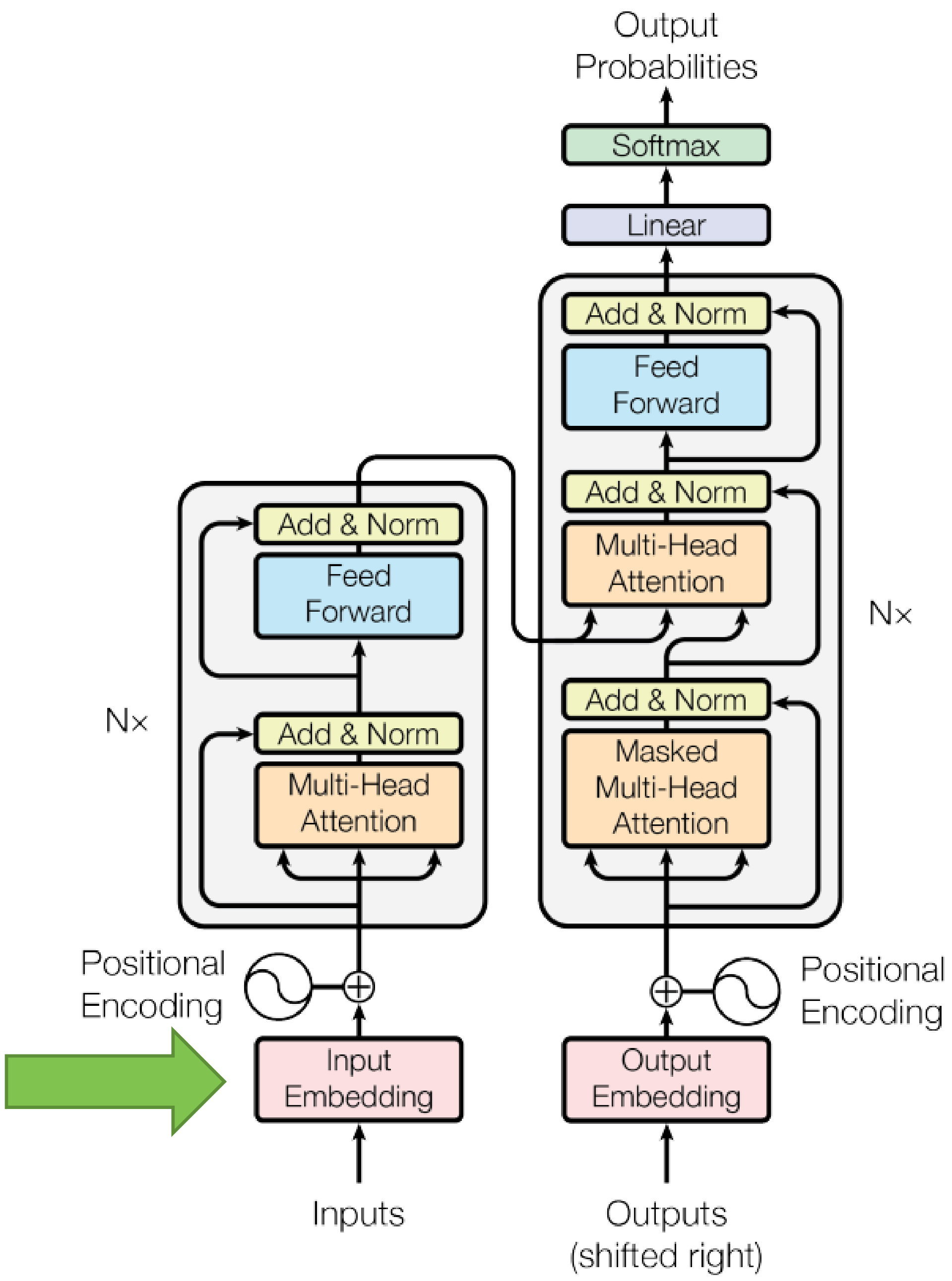
https://arxiv.org/pdf/1706.03762.pdf



Transformer for NLP

Word embedding: Vector representation of words

https://medium.com/@hari4om/word-embedding-d816f643140



Transformer for NLP

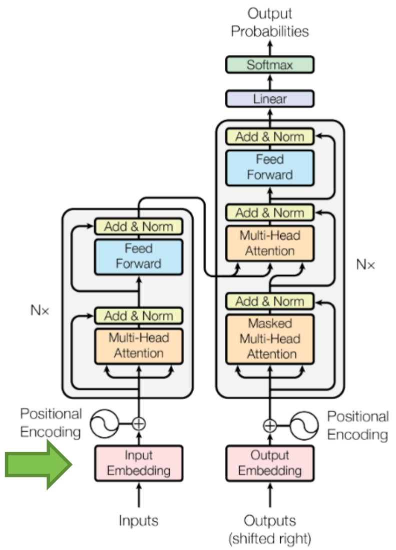
Where there is will, there ....

Each word is assigned an index based on the dictionary of words

Word	Index
a	0
..	..
is	882
..	..
..	..
there	3582
..	..
where	11442
..	..

https://medium.com/@hari4om/word-embedding-d816f643140

Transformer for NLP



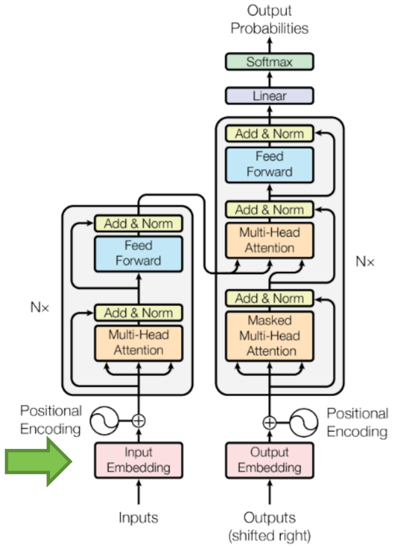
Word	Index
a	0
..	..
is	882
..	..
..	..
there	3582
..	..
where	11442
..	..

Where there is will, there …….

(11442 3582 882 … 3582)

https://medium.com/@hari4om/word-embedding-d816f643140

Transformer for NLP



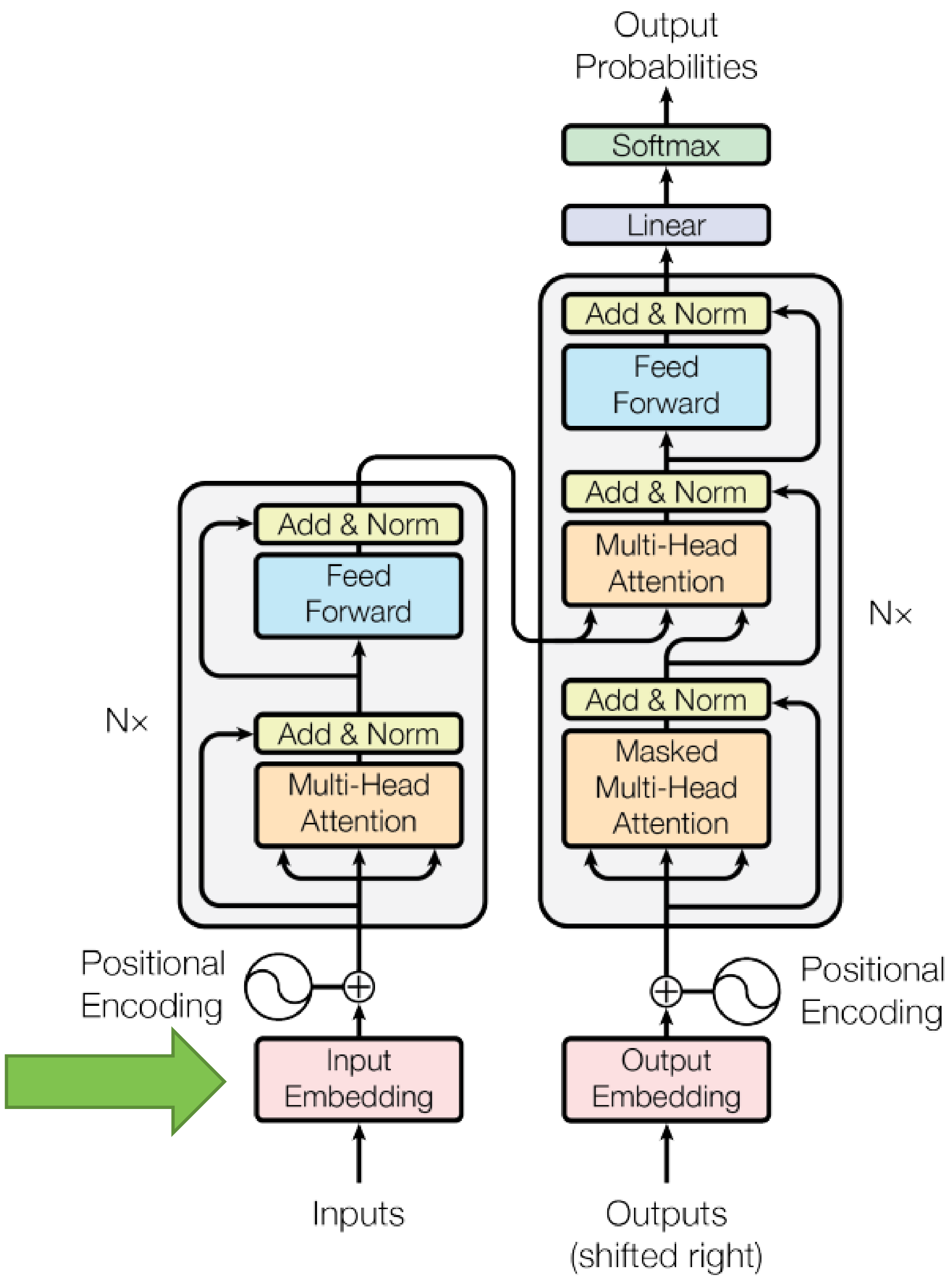
Word	Index
a	0
..	..
is	882
..	..
..	..
there	3582
..	..
where	11442
..	..

Where there is will, there …….

(11442 3582 882 … 3582)

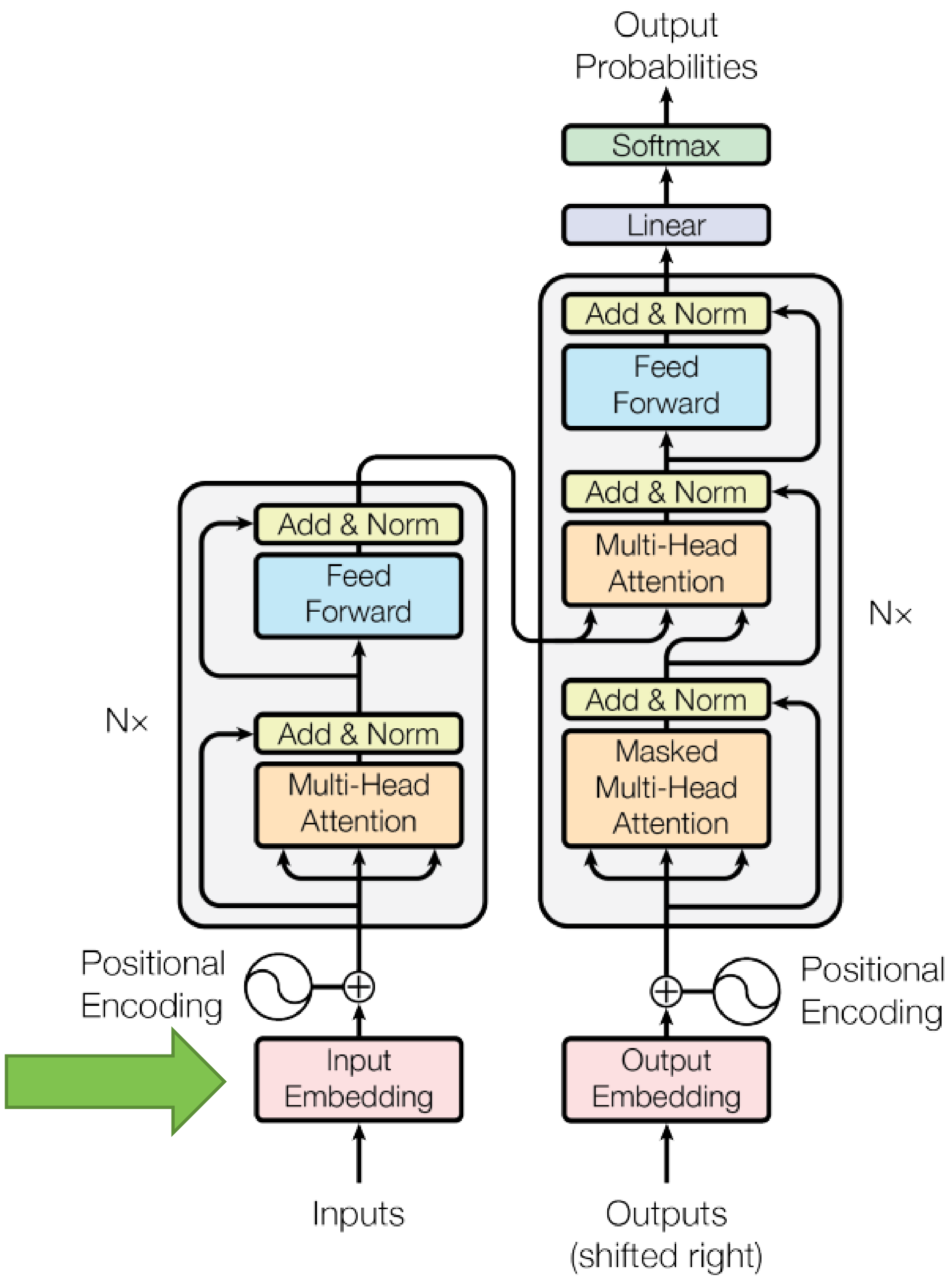
Input

https://medium.com/@hari4om/word-embedding-d816f643140



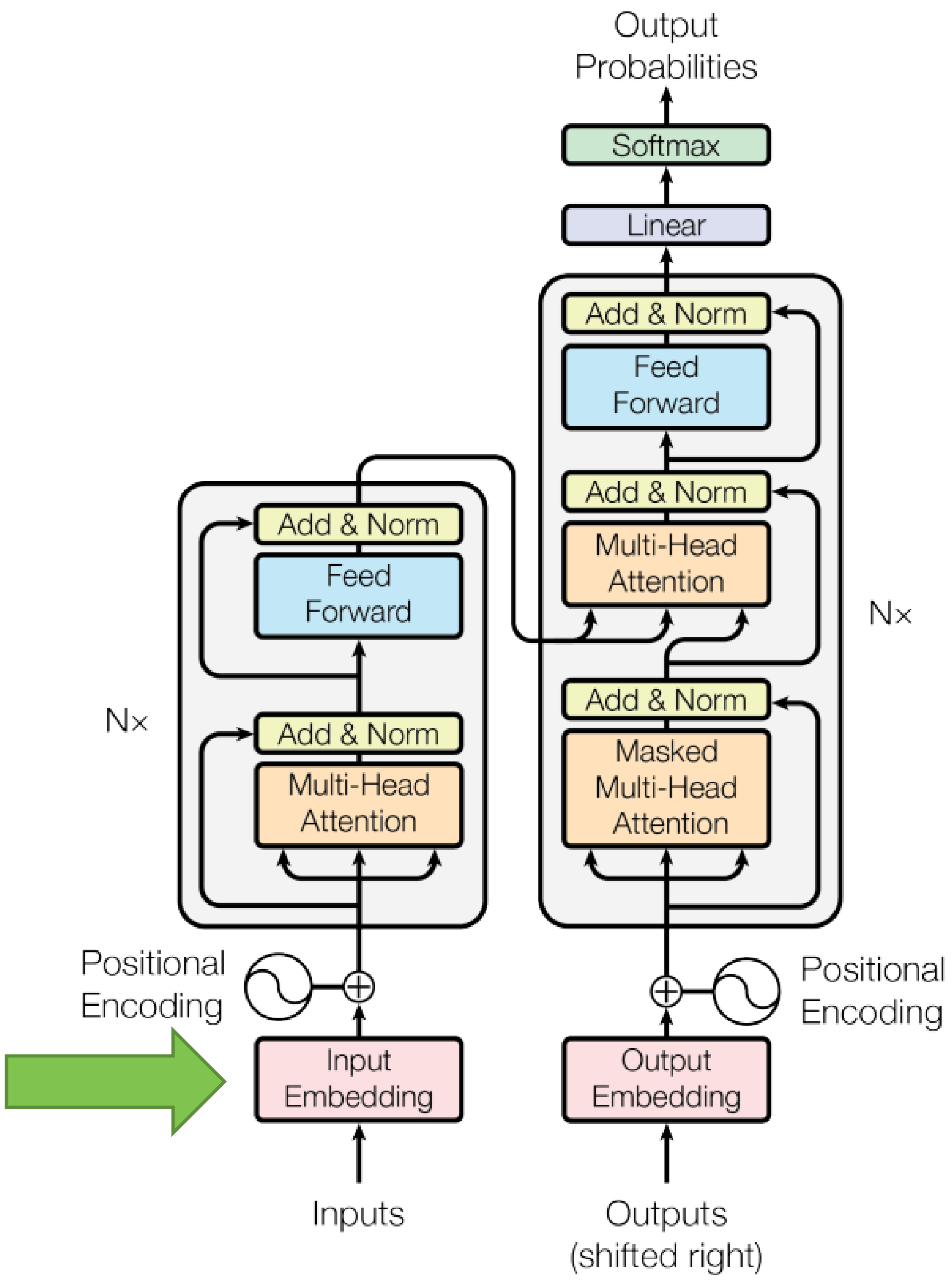
Word Embedding

	living being	feline	human	gender	royalty	verb	plural
cat →	0.6	0.9	0.1	0.4	−0.7	−0.3	−0.2
kitten →	0.5	0.8	−0.1	0.2	−0.6	−0.5	−0.1
dog →	0.7	−0.1	0.4	0.3	−0.4	−0.1	−0.3
houses →	−0.8	−0.4	−0.5	0.1	−0.9	0.3	0.8
man →	0.6	−0.2	0.8	0.9	−0.1	−0.9	−0.7
woman →	0.7	0.3	0.9	−0.7	0.1	−0.5	−0.4
king →	0.5	−0.4	0.7	0.8	0.9	−0.7	−0.6
queen →	0.8	−0.1	0.8	−0.9	0.8	−0.5	−0.9



Word Embedding

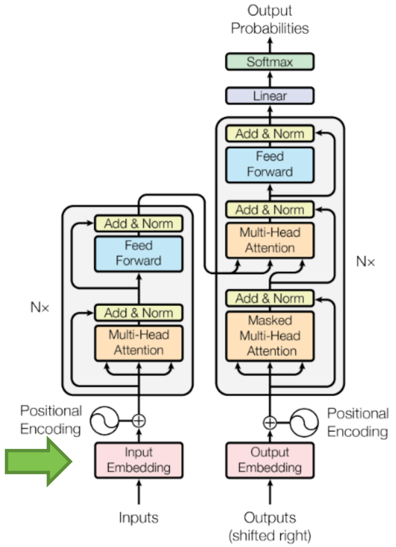
Word embedding: Learnt during training for transformer



Word Embedding

	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7
cat →	0.6	0.9	0.1	0.4	−0.7	−0.3	−0.2
kitten →	0.5	0.8	−0.1	0.2	−0.6	−0.5	−0.1
dog →	0.7	−0.1	0.4	0.3	−0.4	−0.1	−0.3
houses →	−0.8	−0.4	−0.5	0.1	−0.9	0.3	0.8
man →	0.6	−0.2	0.8	0.9	−0.1	−0.9	−0.7
woman →	0.7	0.3	0.9	−0.7	0.1	−0.5	−0.4
king →	0.5	−0.4	0.7	0.8	0.9	−0.7	−0.6
queen →	0.8	−0.1	0.8	−0.9	0.8	−0.5	−0.9

After Word Embedding



Word	Index
a	0
..	..
is	882
..	..
..	..
there	3582
..	..
where	11442
..	..

	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7
cat →	0.6	0.9	0.1	0.4	−0.7	−0.3	−0.2
kitten →	0.5	0.8	−0.1	0.2	−0.6	−0.5	−0.1
dog →	0.7	−0.1	0.4	0.3	−0.4	−0.1	−0.3
houses →	−0.8	−0.4	−0.5	0.1	−0.9	0.3	0.8
man →	0.6	−0.2	0.8	0.9	−0.1	−0.9	−0.7
woman →	0.7	0.3	0.9	−0.7	0.1	−0.5	−0.4
king →	0.5	−0.4	0.7	0.8	0.9	−0.7	−0.6
queen →	0.8	−0.1	0.8	−0.9	0.8	−0.5	−0.9

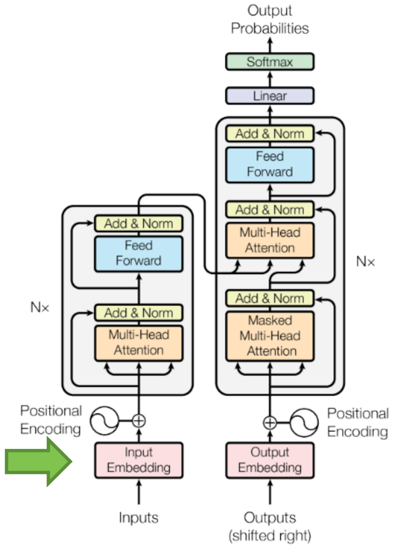
Where there is will, there ....

(11442 3582 882 ... 3582)

0.30	0.17		0.20
0.22	0.29		0.52
0.17	0.10	...	0.18
0.65	0.32		0.69
0.12	0.11		0.72

https://medium.com/@hari4om/word-embedding-d816f643140

After Word Embedding



Word	Index
a	0
..	..
is	882
..	..
..	..
there	3582
..	..
where	11442
..	..

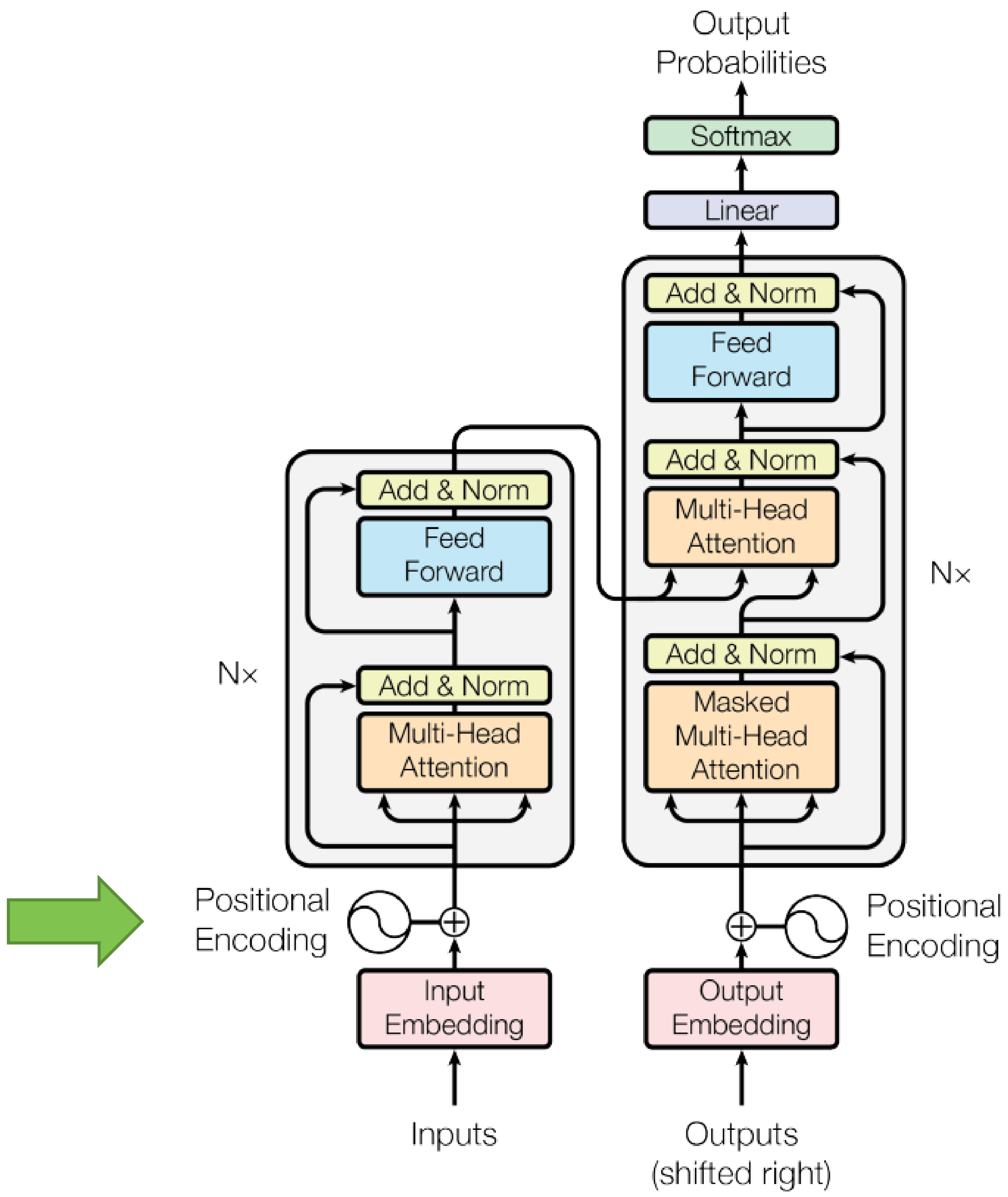
	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	Feature 6	Feature 7
cat →	0.6	0.9	0.1	0.4	−0.7	−0.3	−0.2
kitten →	0.5	0.8	−0.1	0.2	−0.6	−0.5	−0.1
dog →	0.7	−0.1	0.4	0.3	−0.4	−0.1	−0.3
houses →	−0.8	−0.4	−0.5	0.1	−0.9	0.3	0.8
man →	0.6	−0.2	0.8	0.9	−0.1	−0.9	−0.7
woman →	0.7	0.3	0.9	−0.7	0.1	−0.5	−0.4
king →	0.5	−0.4	0.7	0.8	0.9	−0.7	−0.6
queen →	0.8	−0.1	0.8	−0.9	0.8	−0.5	−0.9

Where there is will, there ....

(11442 3582 882 ... 3582)

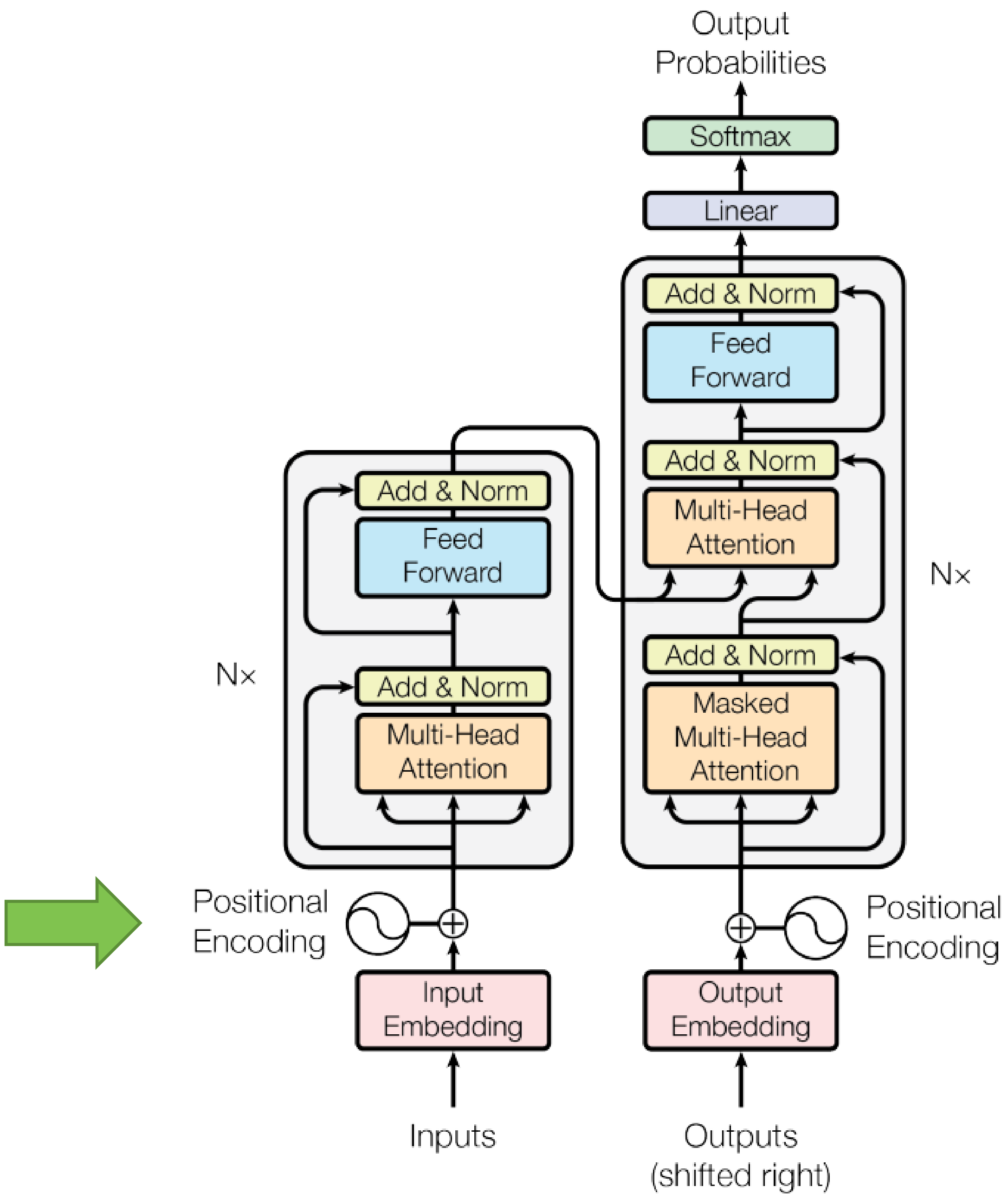
e_0	e_1	...	e_n
0.30	0.17		0.20
0.22	0.29		0.52
0.17	0.10		0.18
0.65	0.32		0.69
0.12	0.11		0.72

https://medium.com/@hari4om/word-embedding-d816f643140



Positional Embedding

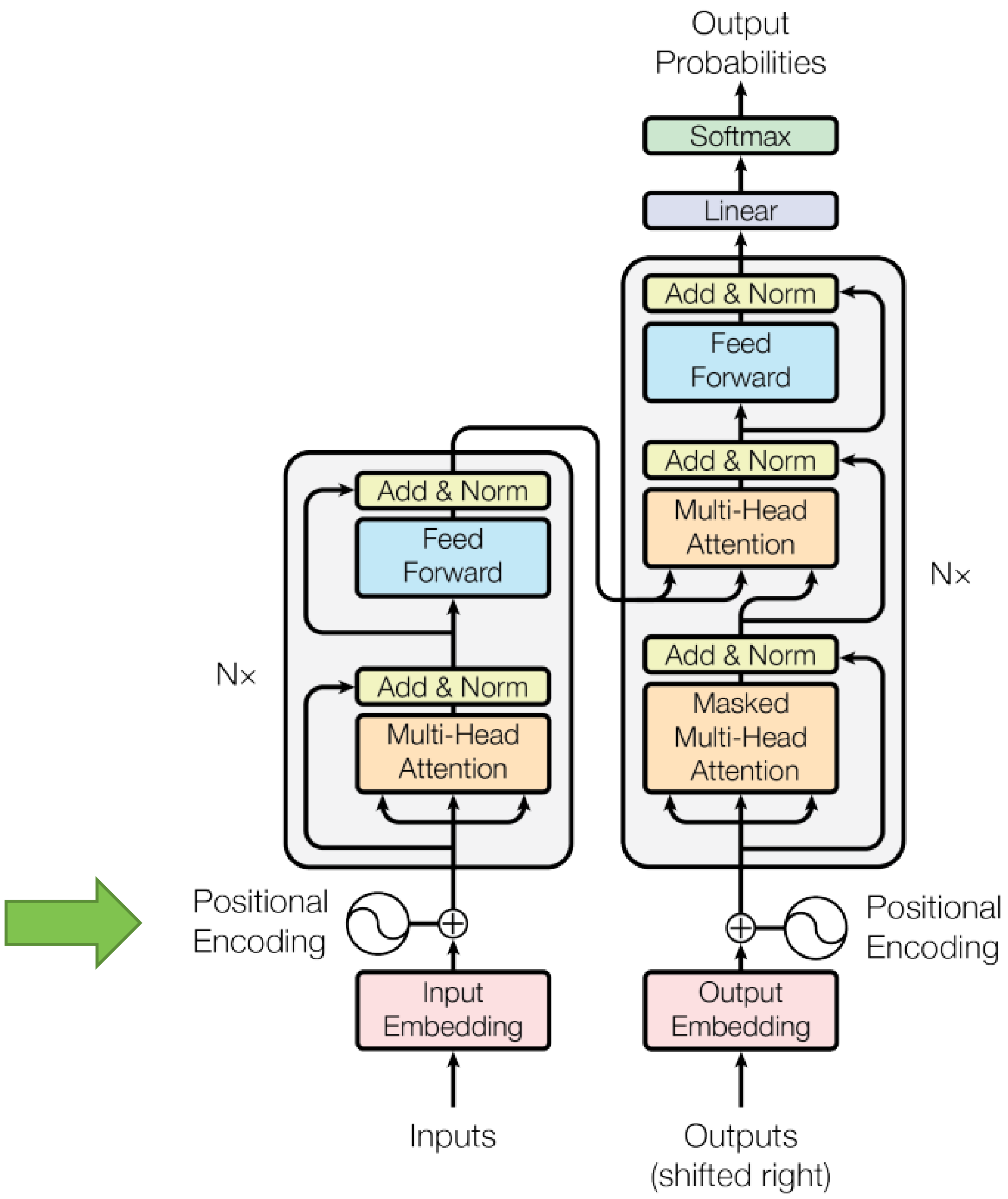
Although the sky is cloudy, it may not rain



Positional Embedding

Although the sky is cloudy, it may not rain

Although the sky is not cloudy, it may rain



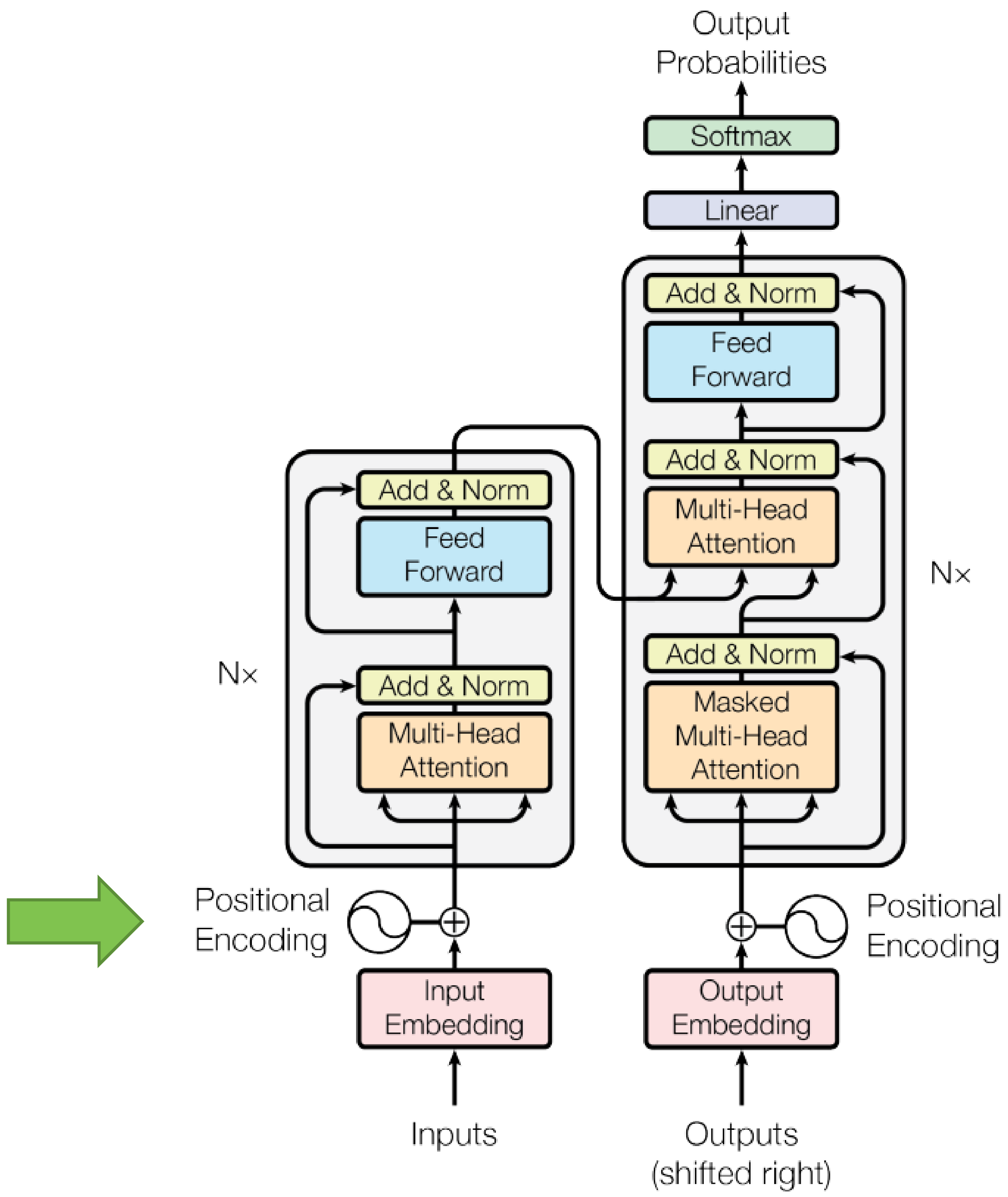
Positional Embedding

Although the sky is cloudy, it may not rain

Although the sky is not cloudy, it may rain

The position of the word ‘not’ changes
the meaning of the sentence

Positional Embedding

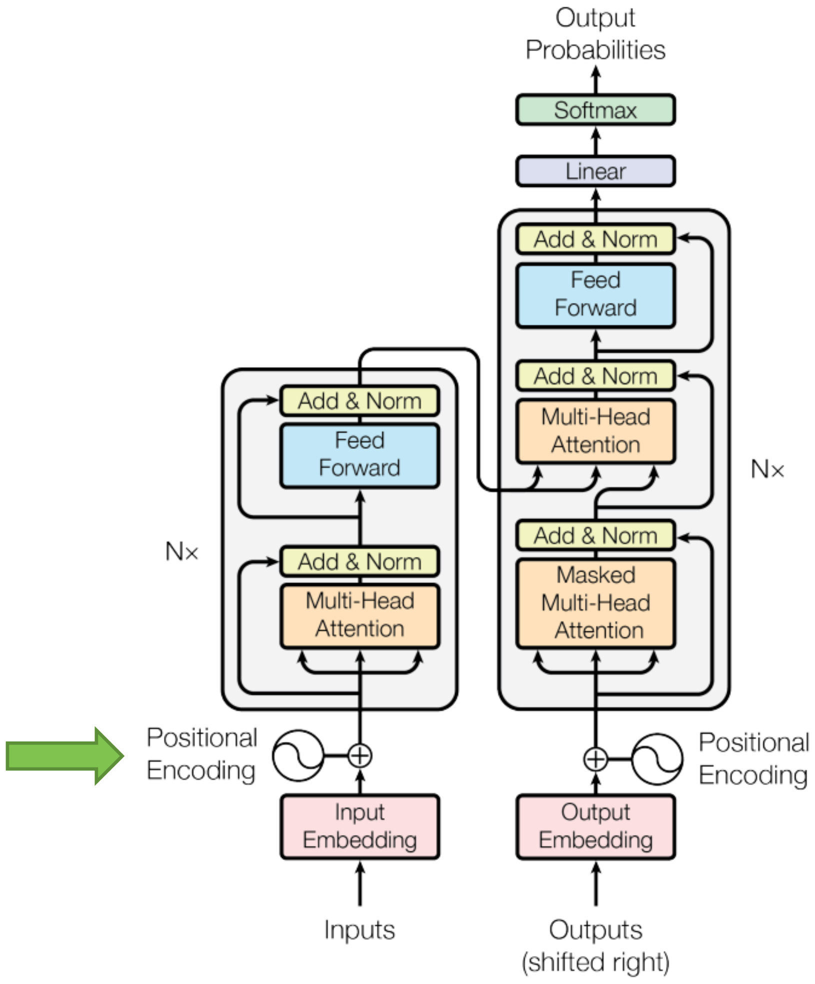


Although the sky is cloudy, it may not rain

Although the sky is not cloudy, it may rain

The position of the word 'not' changes the meaning of the sentence

So, position information should be incorporated with word embedding

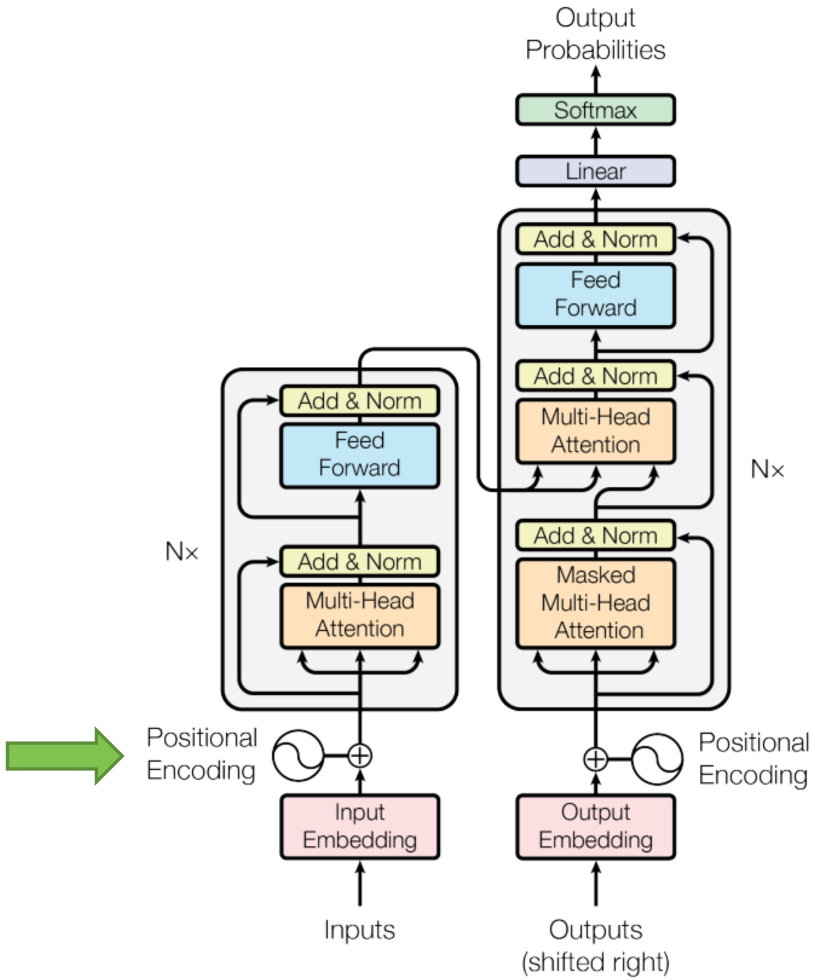


Positional Embedding

Where there is will, there …….

(11442	3582	882 …	3582)
0.30	0.17		0.20
0.22	0.29		0.52
0.17	0.10	…	0.18
0.65	0.32		0.69
0.12	0.11		0.72
e_0	e_1		e_n

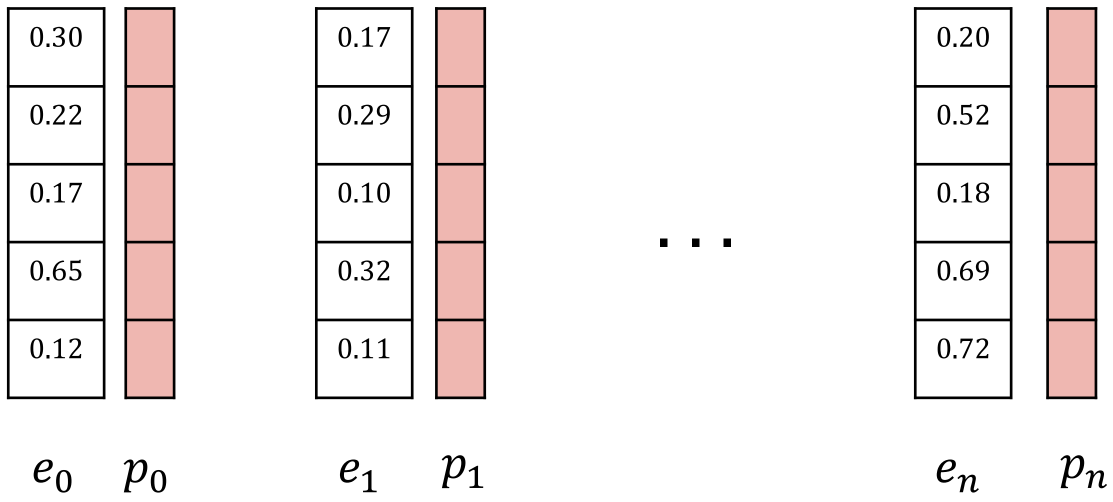
https://medium.com/@hari4om/word-embedding-d816f643140



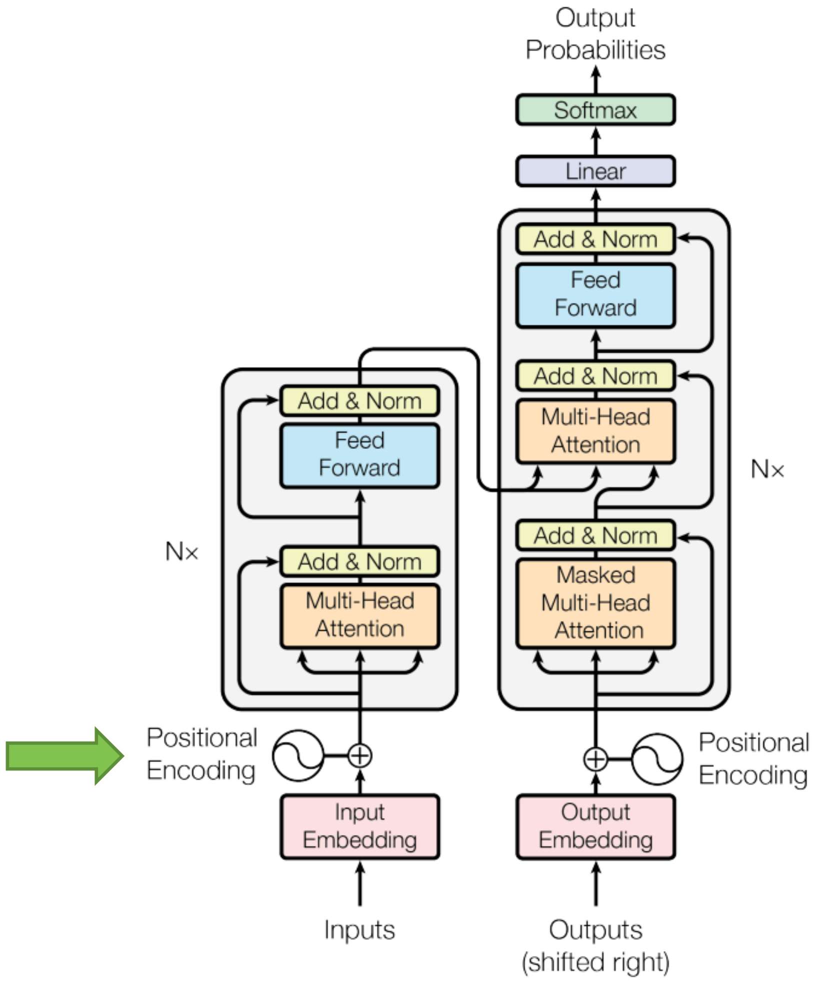
Positional Embedding

Where there is will, there …….

(11442 3582 882 … 3582)



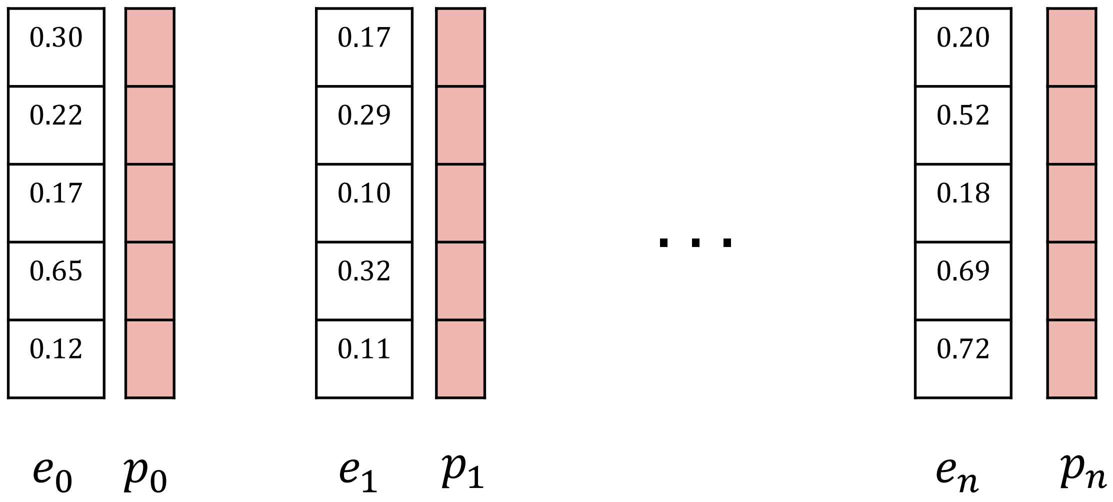
https://medium.com/@hari4om/word-embedding-d816f643140



Positional Embedding

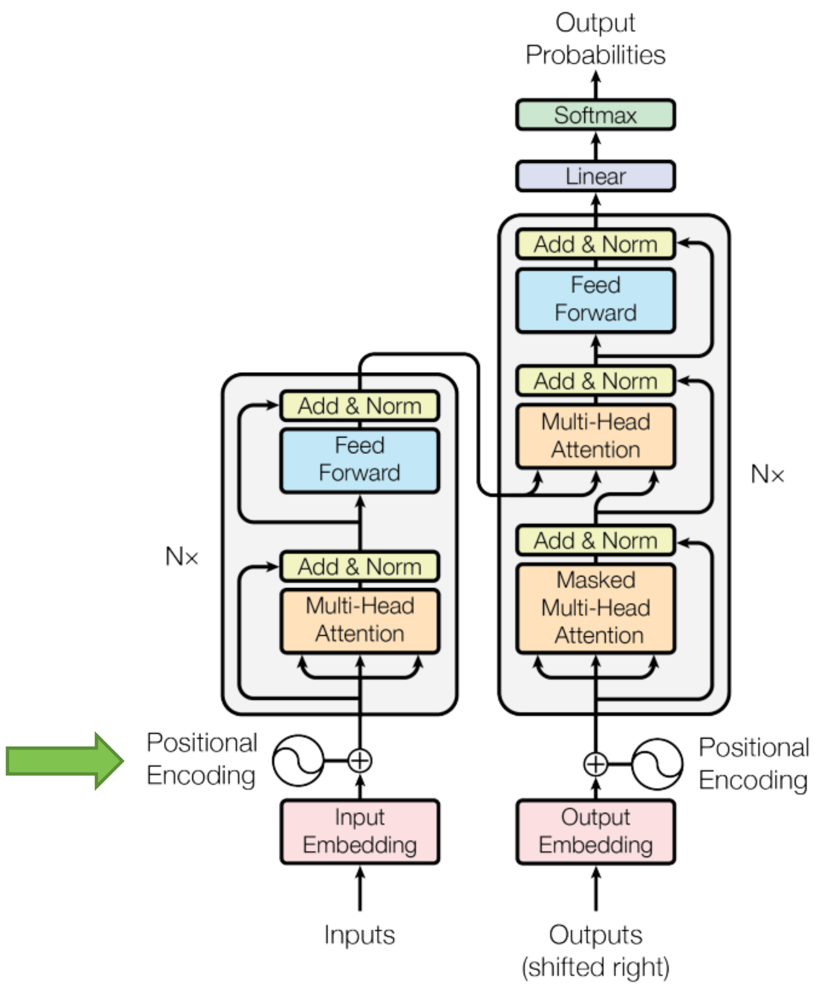
Where there is will, there …….

(11442 3582 882 … 3582)



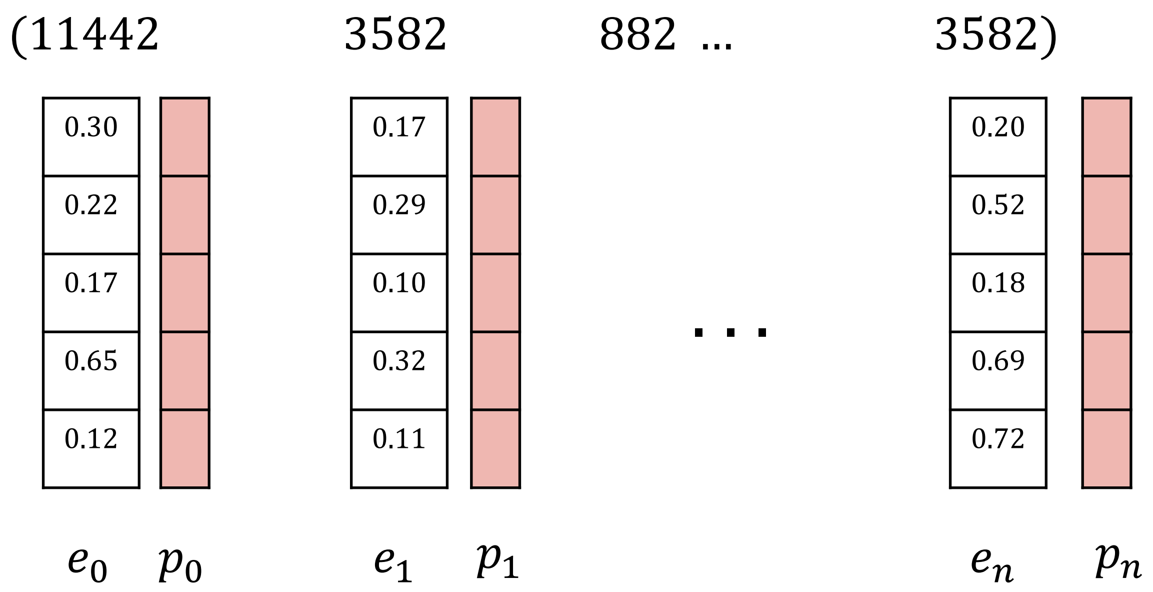
How to get p_i?

https://medium.com/@hari4om/word-embedding-d816f643140



Positional Embedding

Where there is will, there ....

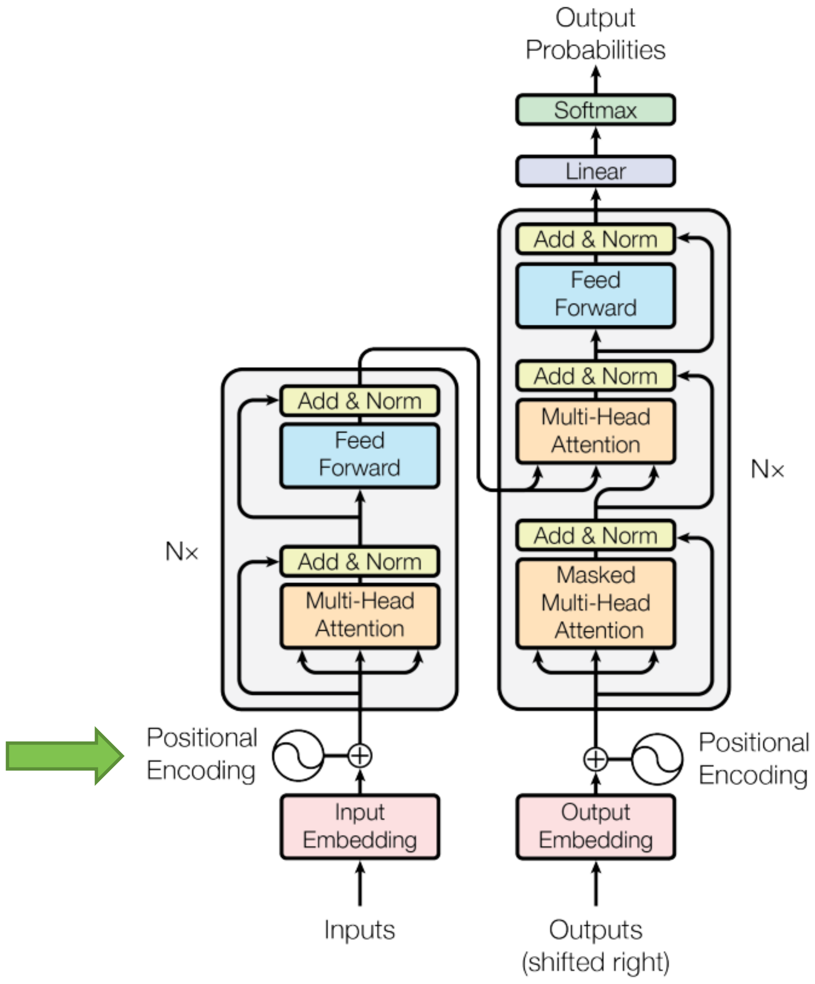


How to get p_i? Vaswani *et al.*

$$PE(pos, 2i) = \sin\left(\frac{pos}{1000^{\frac{2i}{d}}}\right)$$ Dimesnions at even positions

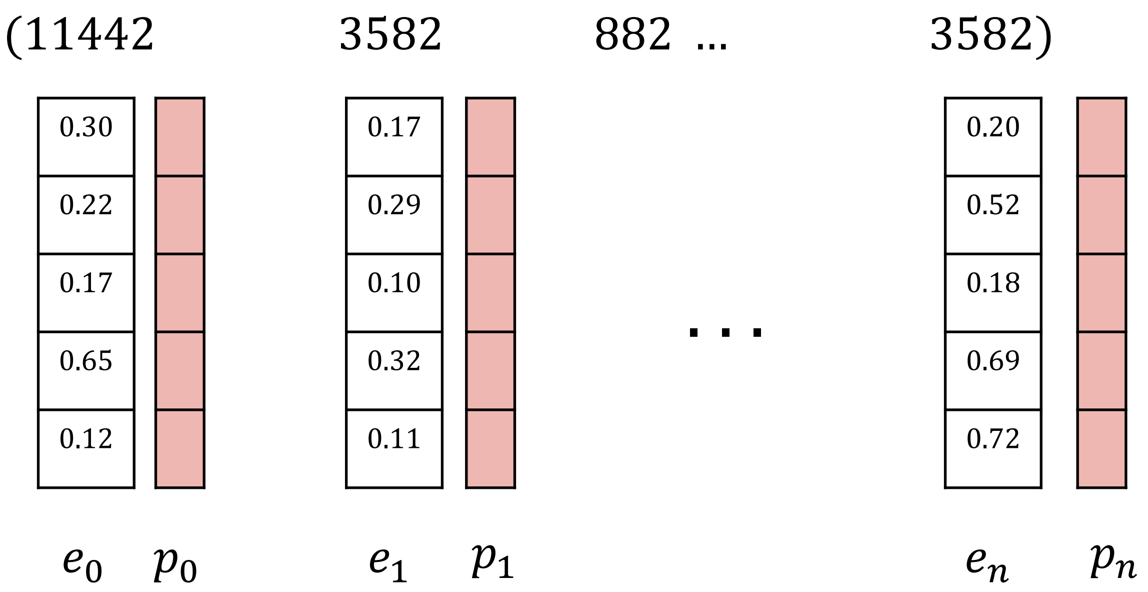
$$PE(pos, 2i+1) = \cos\left(\frac{pos}{1000^{\frac{2i}{d}}}\right)$$ Dimensions at odd positions

https://medium.com/@hari4om/word-embedding-d816f643140



Positional Embedding

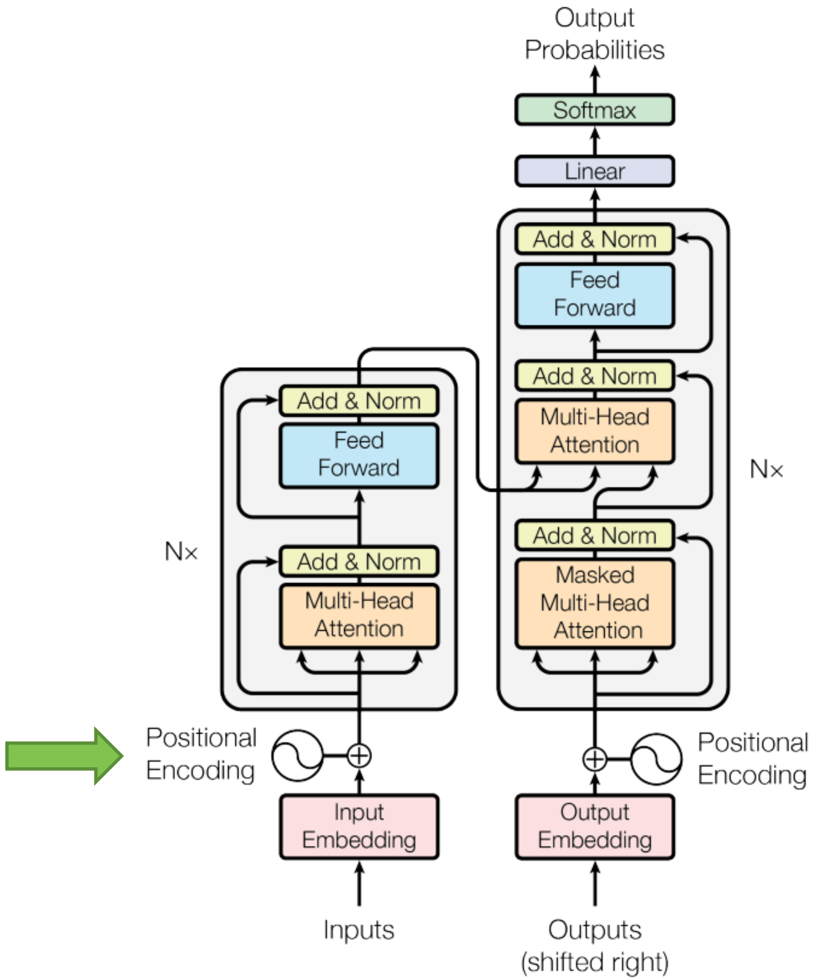
Where there is will, there ....



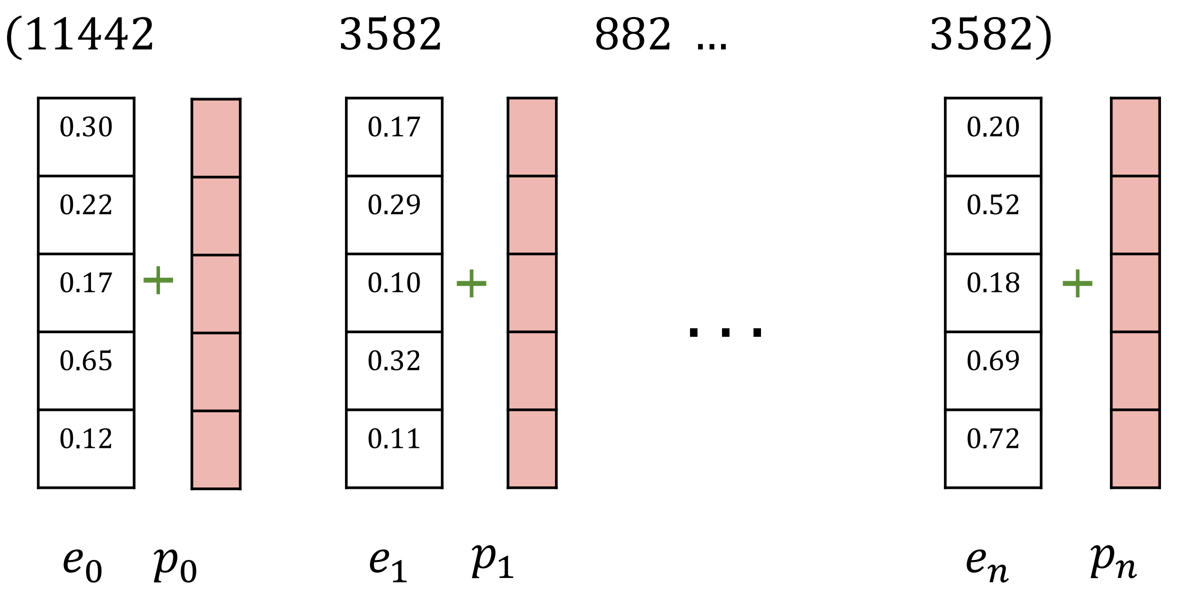
How to combine e_i and p_i?

https://medium.com/@hari4om/word-embedding-d816f643140

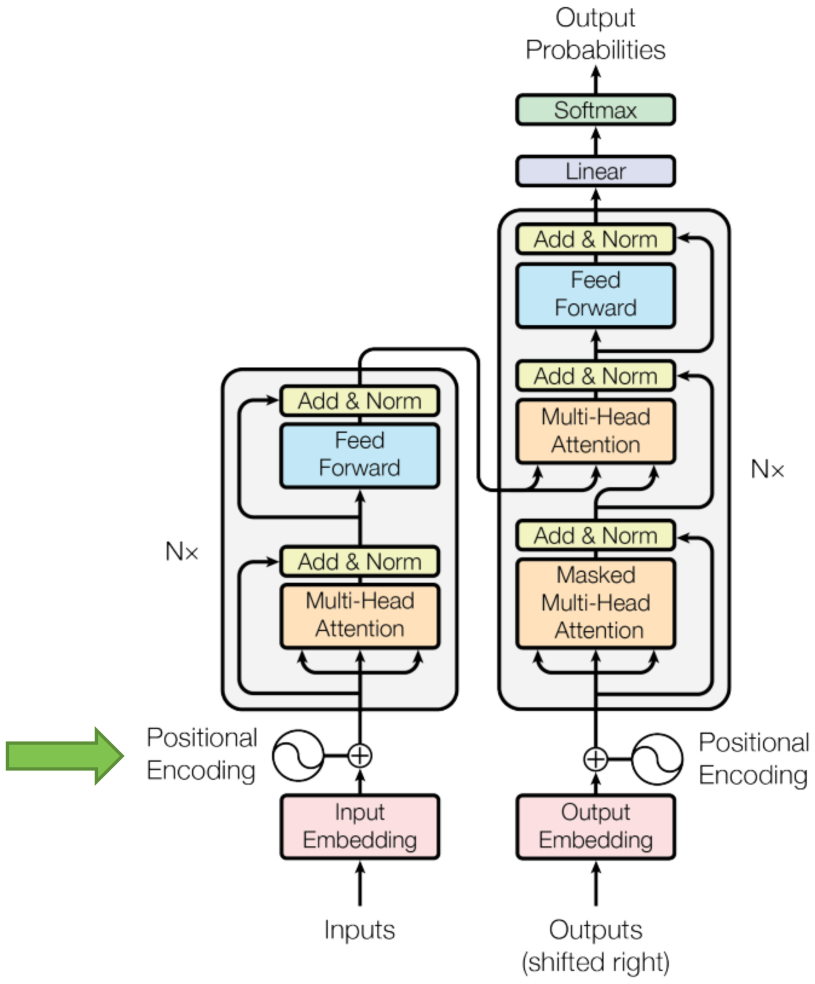
Positional Embedding



Where there is will, there ....



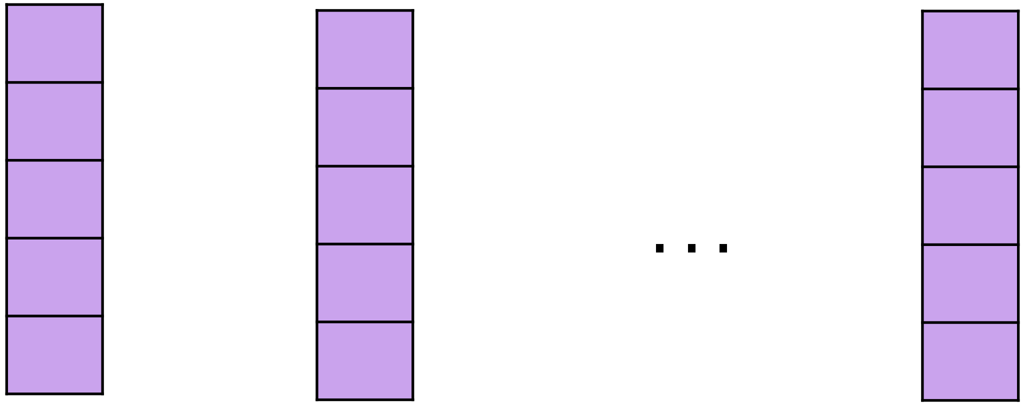
https://medium.com/@hari4om/word-embedding-d816f643140



Positional Embedding

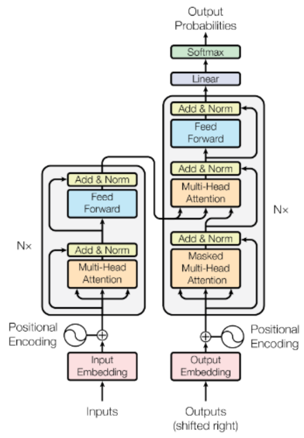
Where there is will, there ....

(11442 3582 882 ... 3582)



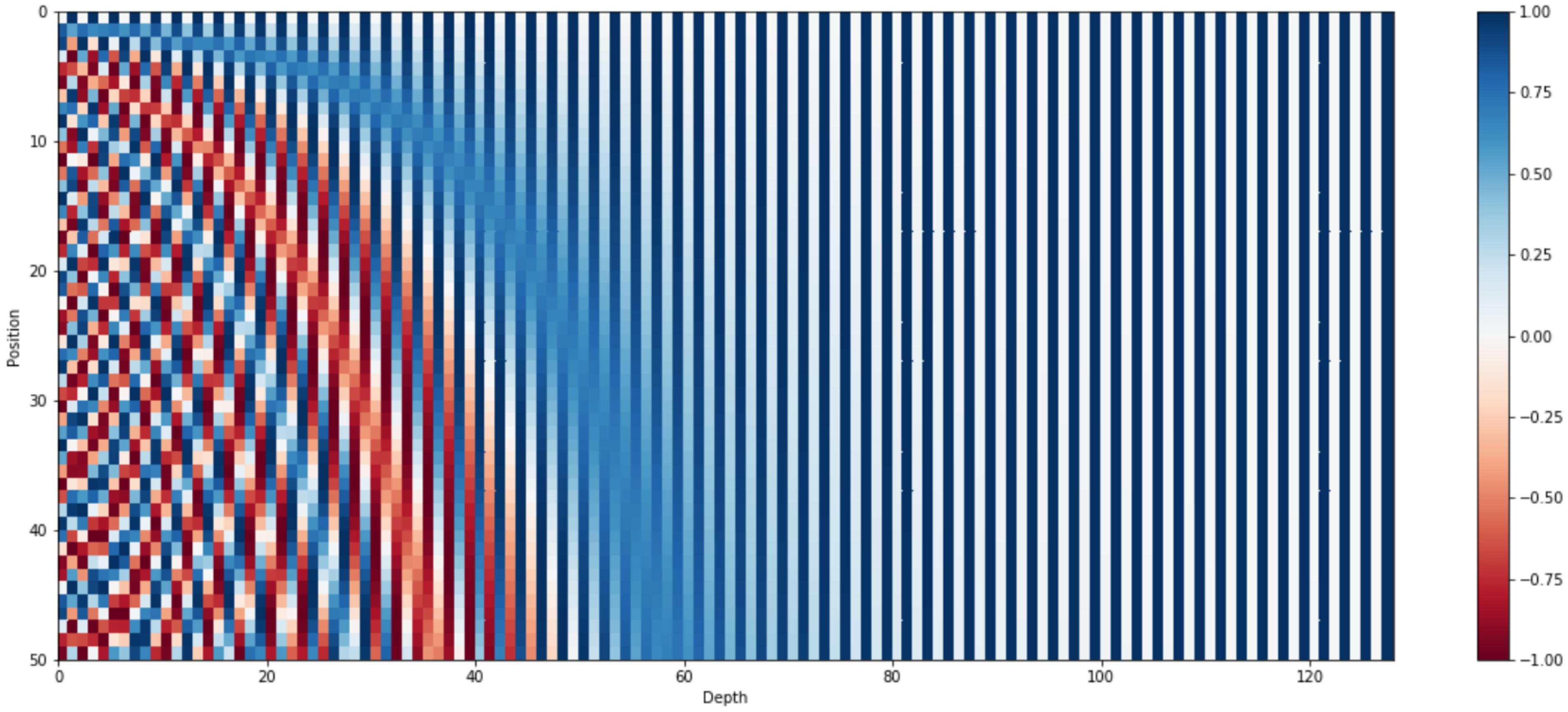
ep_0 ep_1 ep_n

https://medium.com/@hari4om/word-embedding-d816f643140



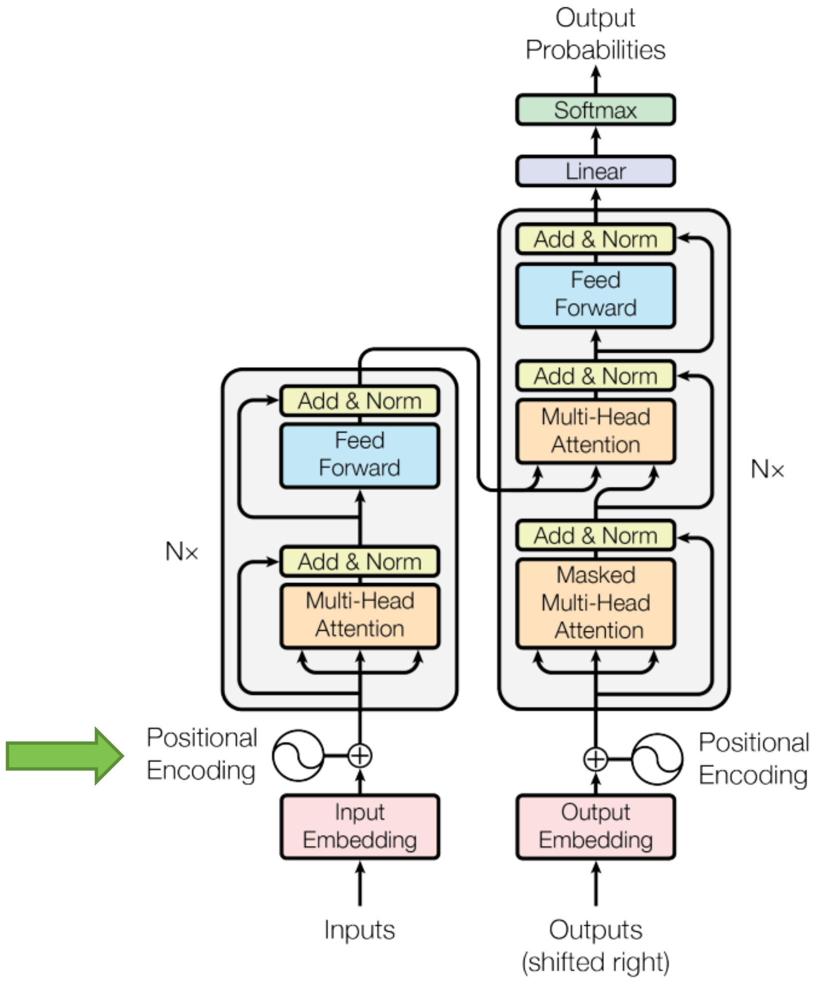
Positional Embedding

Where there is will, there …….



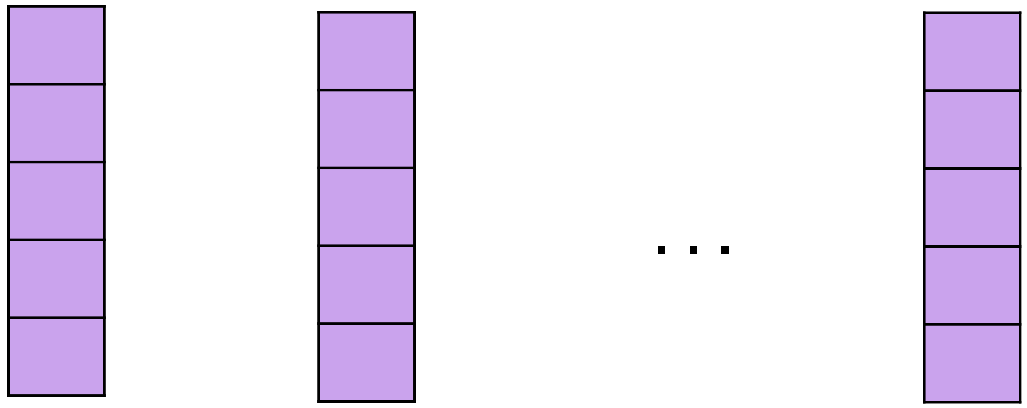
https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Positional Embedding



Where there is will, there …….

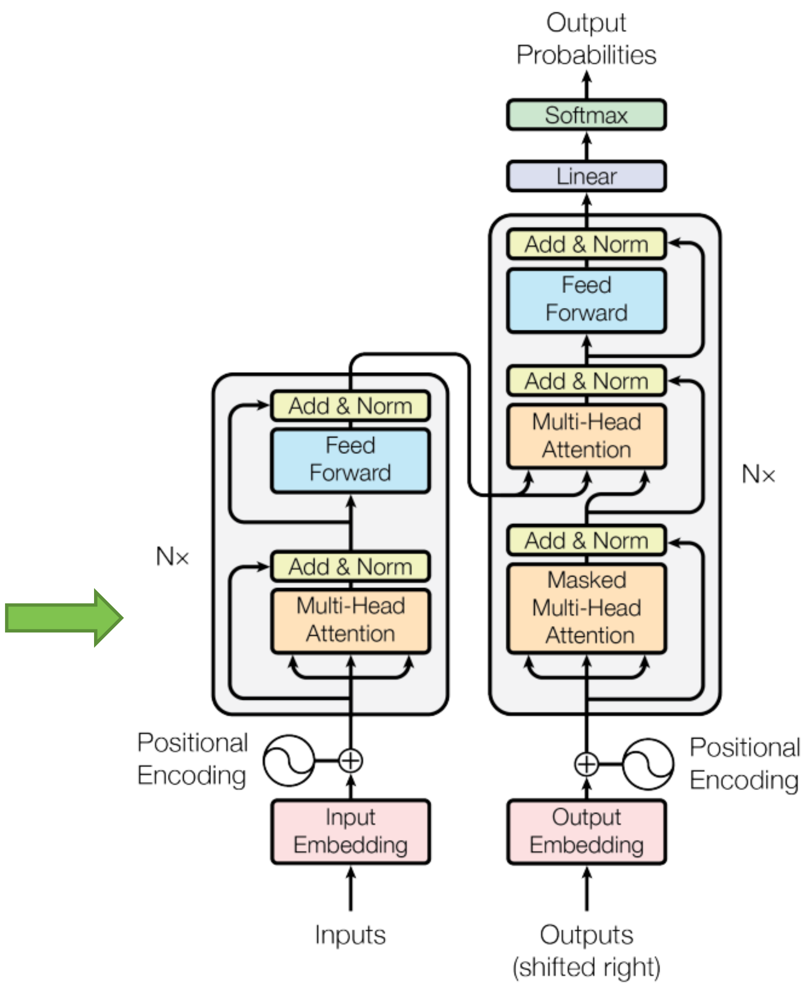
(11442 3582 882 … 3582)

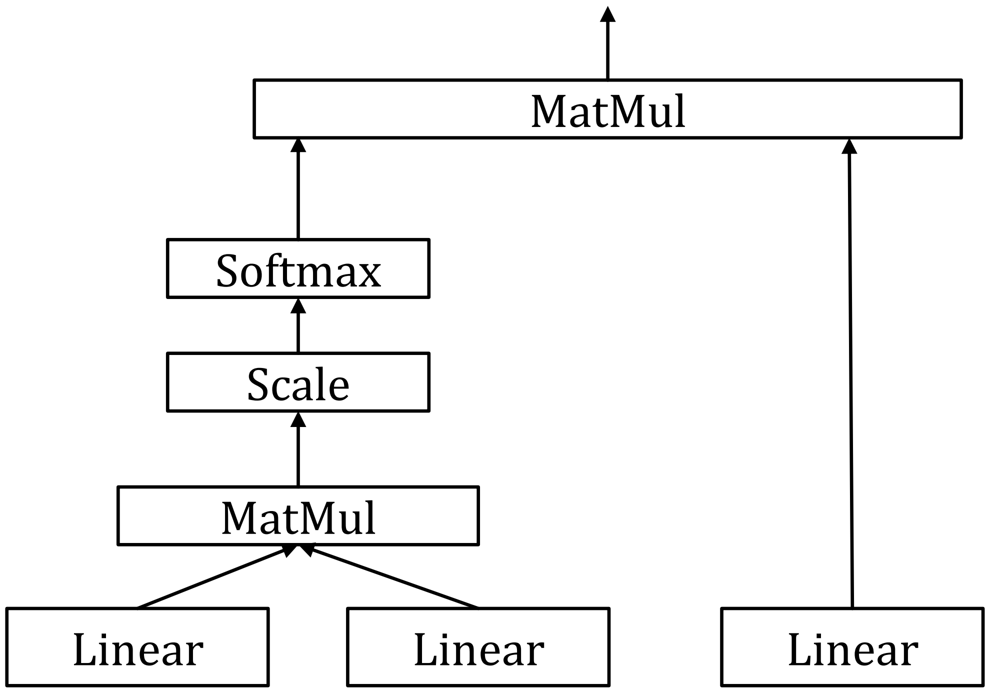


ep_0 ep_1 ep_n

https://medium.com/@hari4om/word-embedding-d816f643140

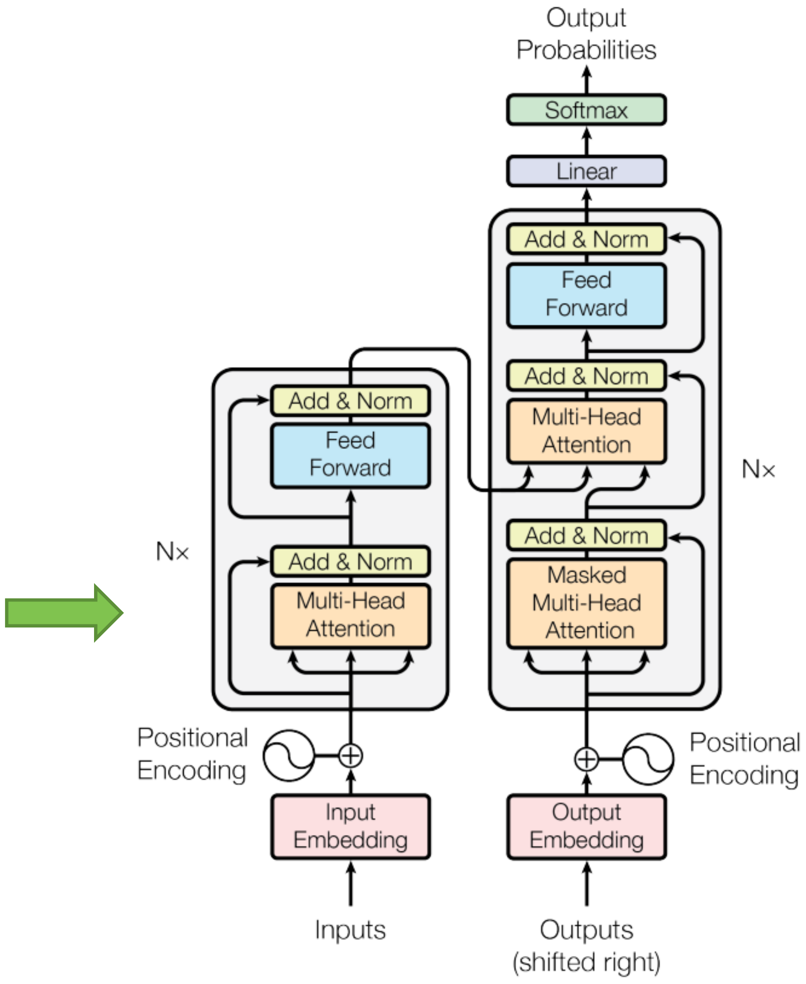
Multi-head Attention

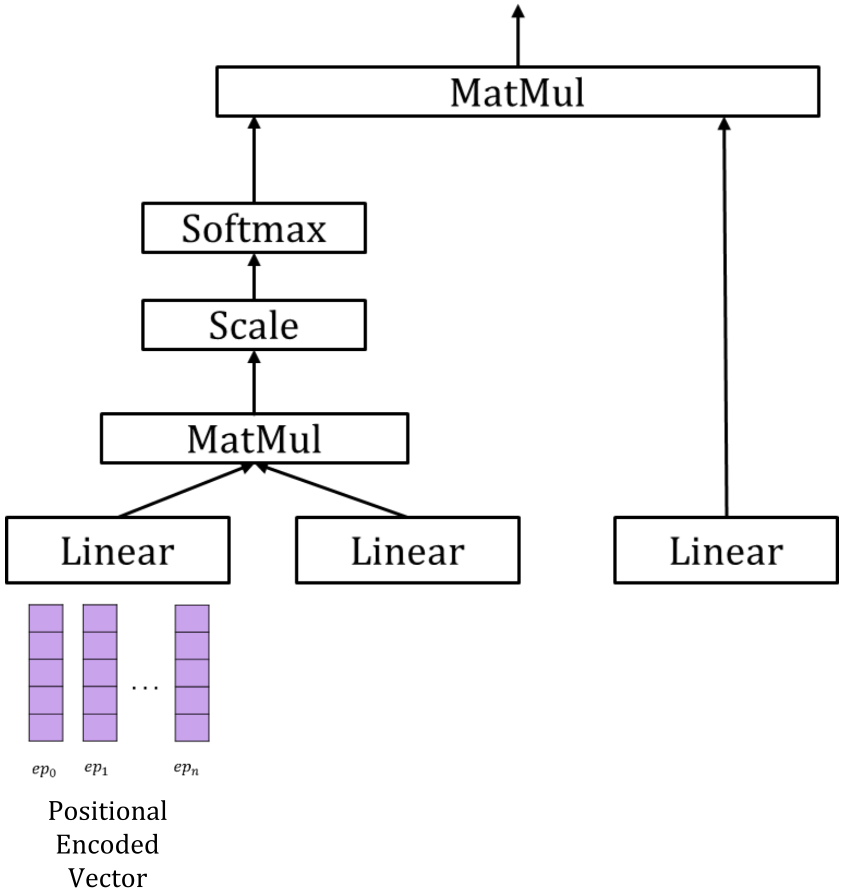




https://medium.com/@hari4om/word-embedding-d816f643140

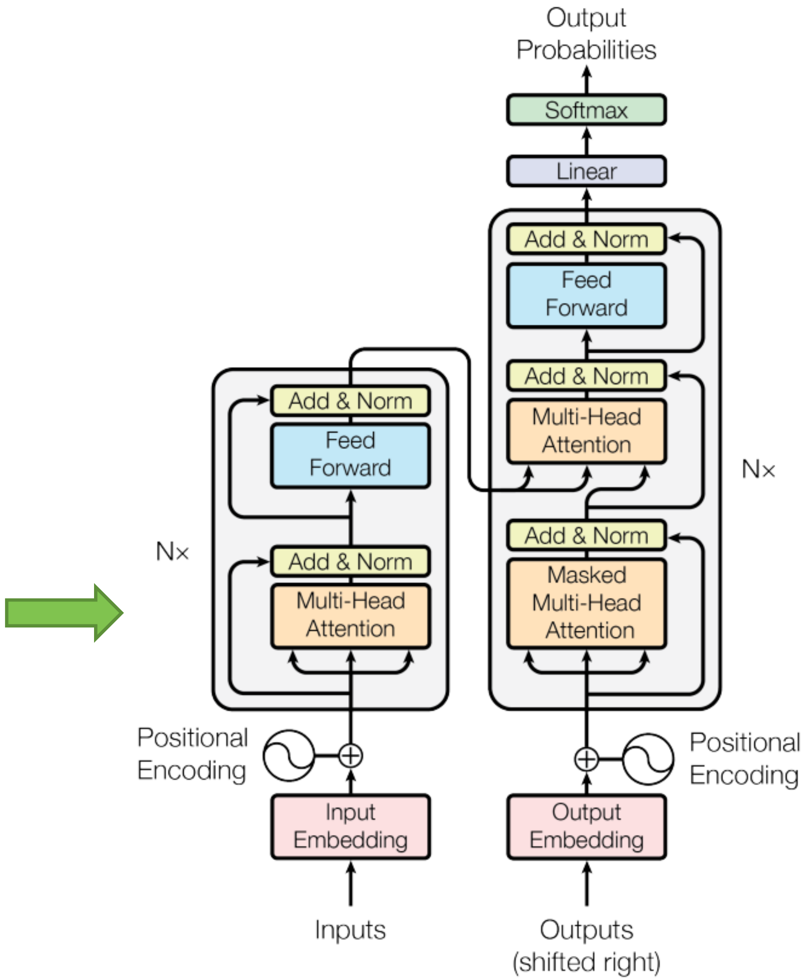
Multi-head Attention

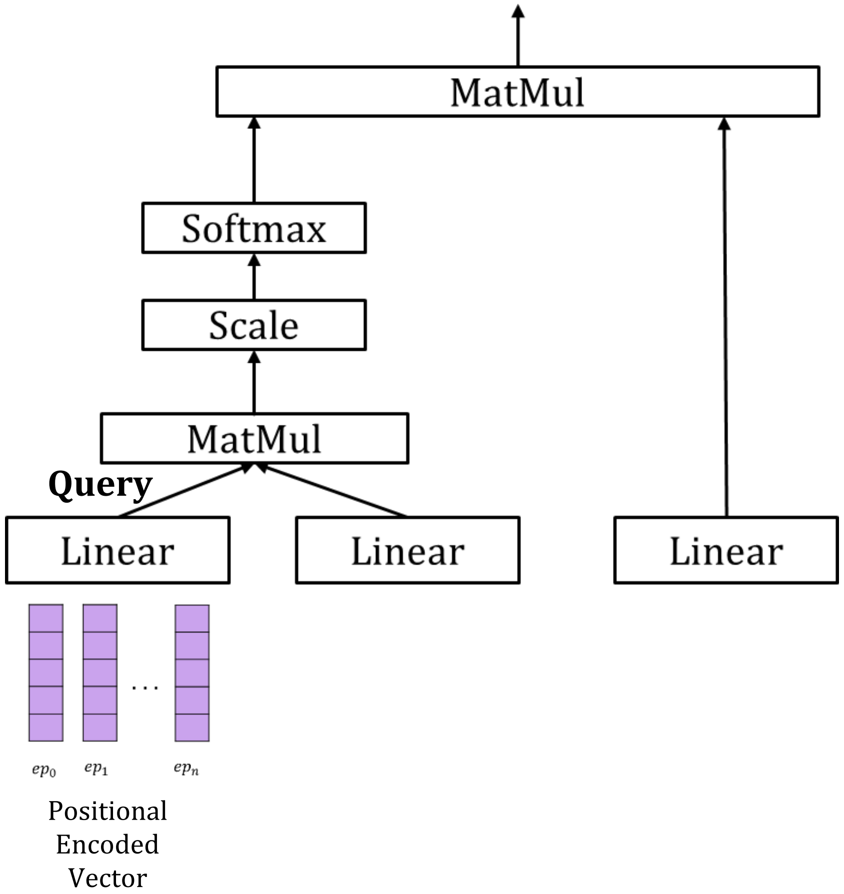




https://medium.com/@hari4om/word-embedding-d816f643140

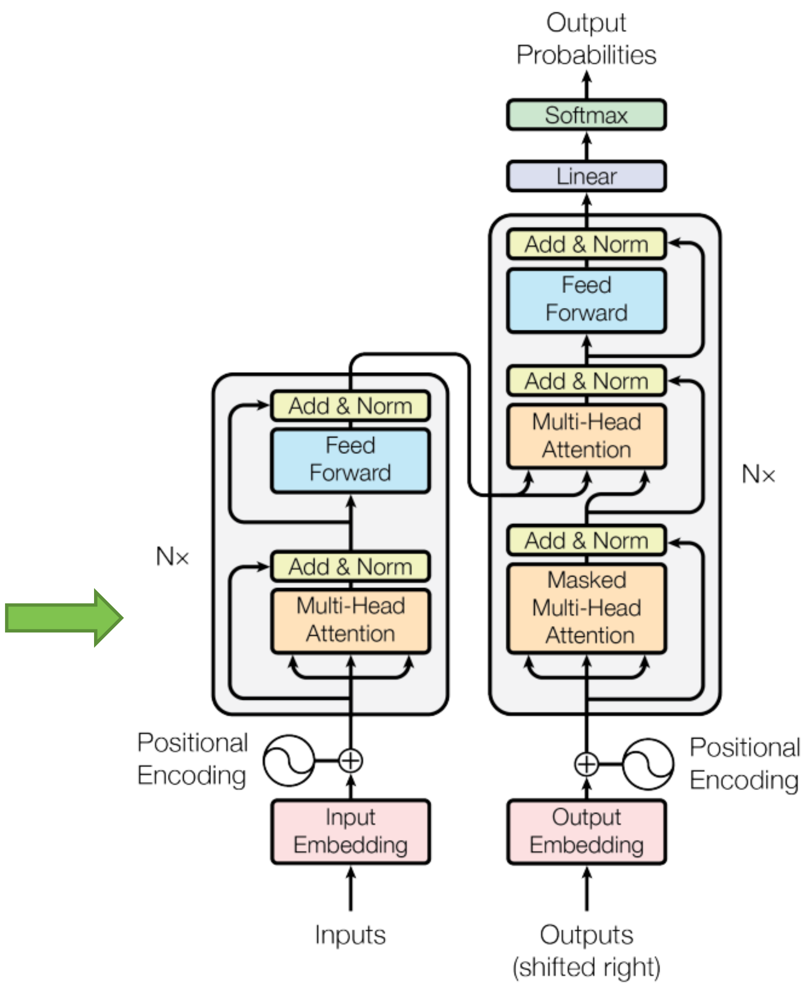
Multi-head Attention

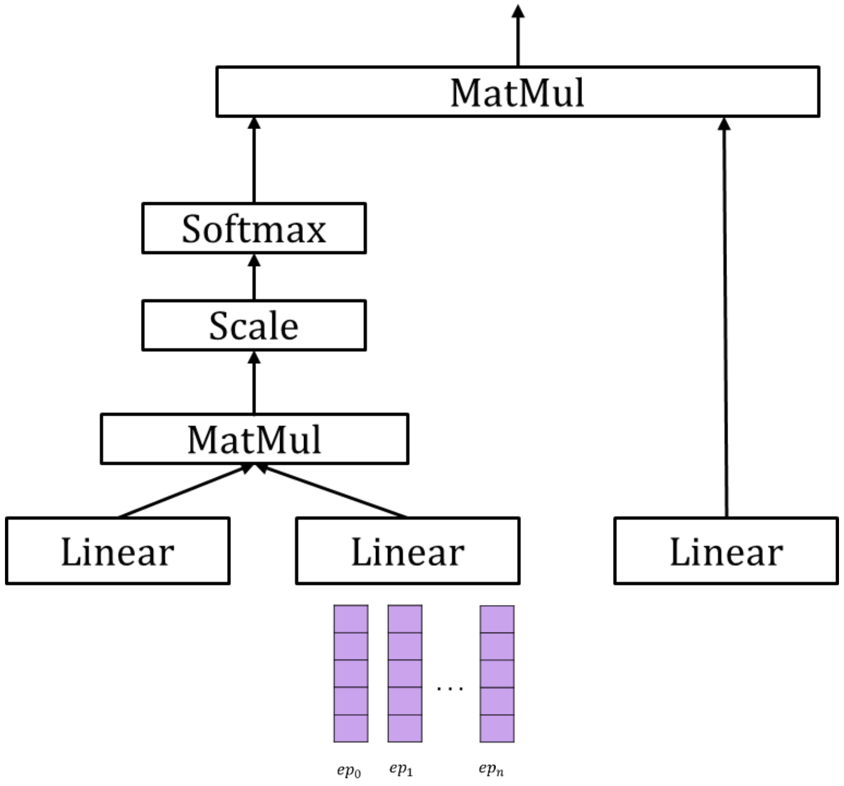




https://medium.com/@hari4om/word-embedding-d816f643140

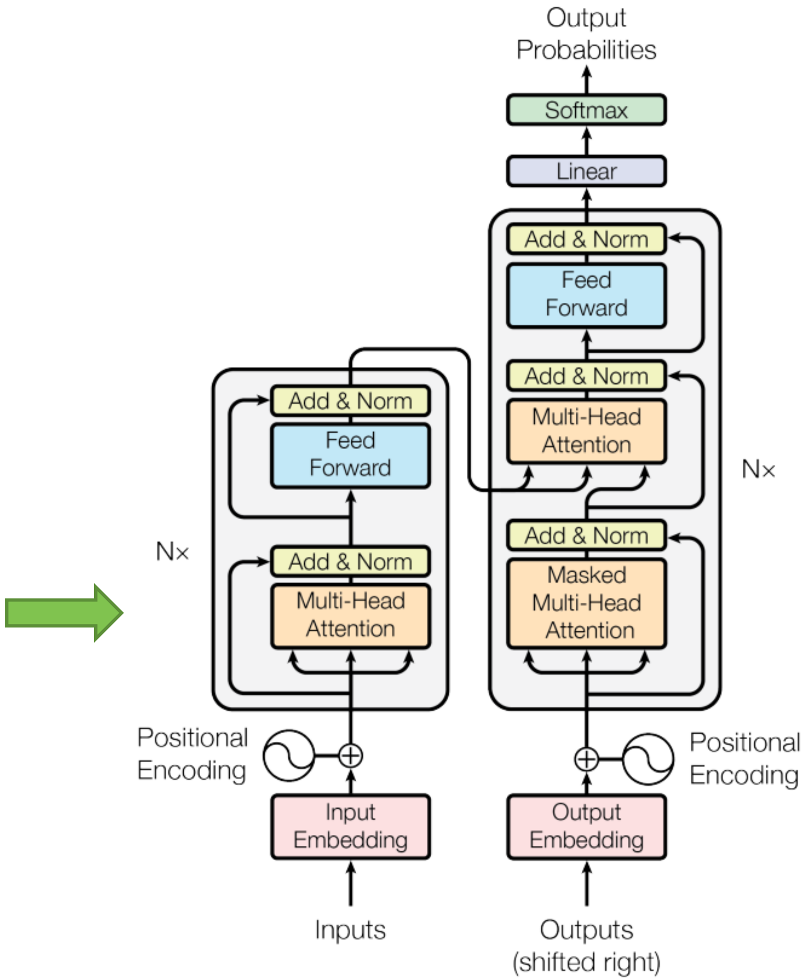
Multi-head Attention

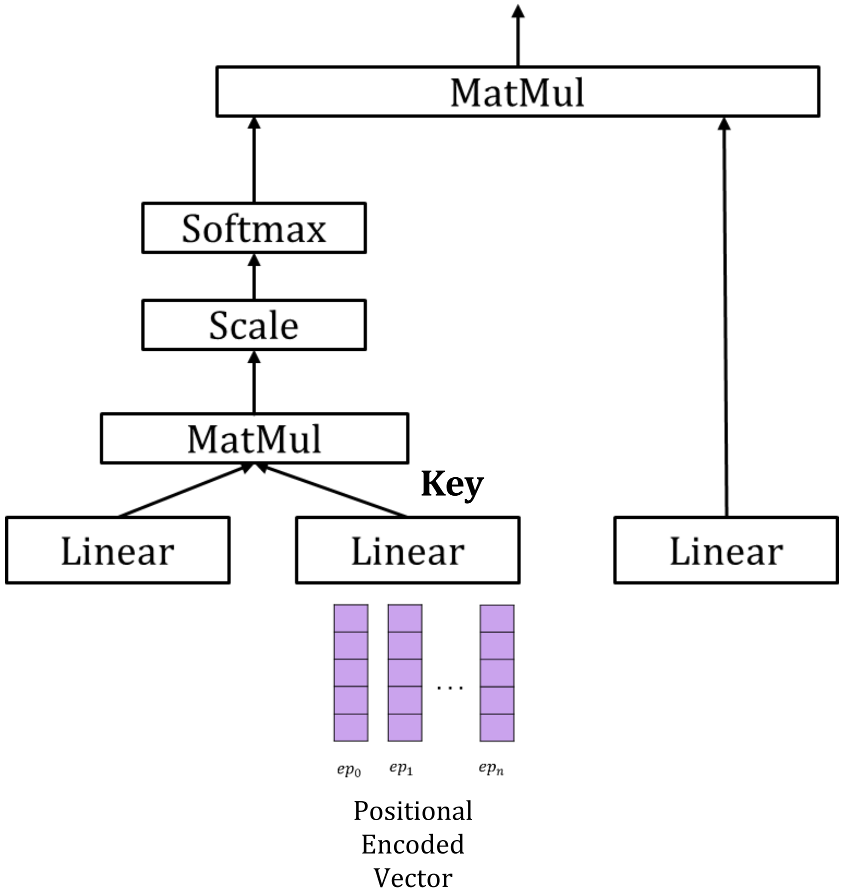




https://medium.com/@hari4om/word-embedding-d816f643140

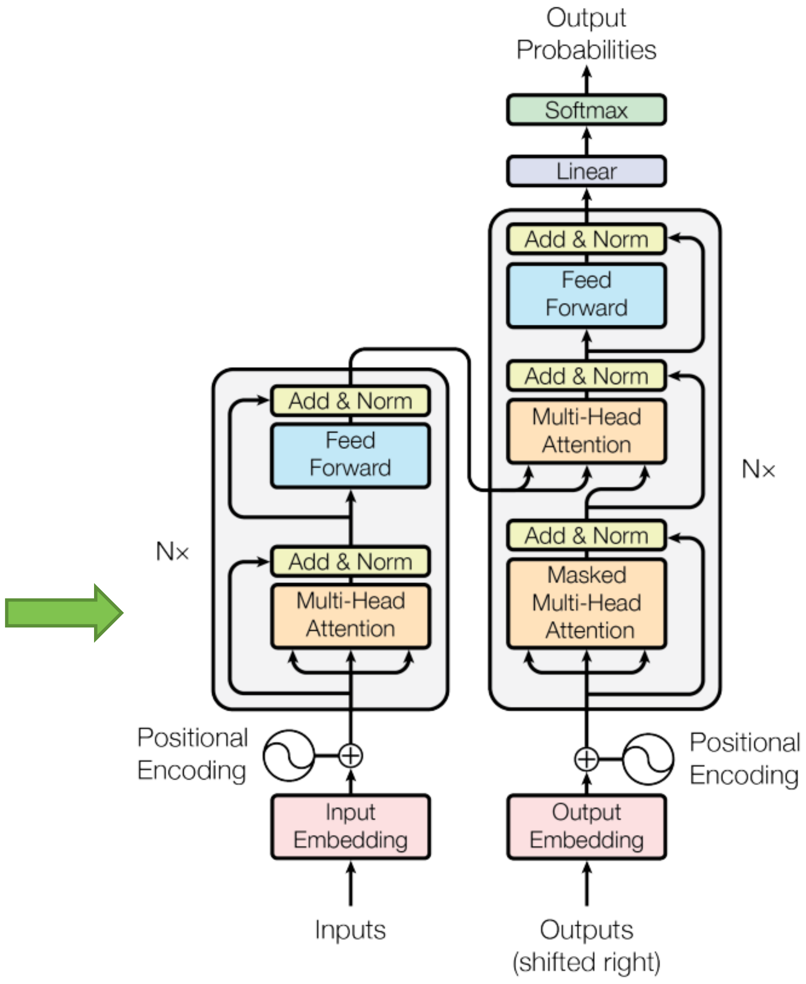
Multi-head Attention

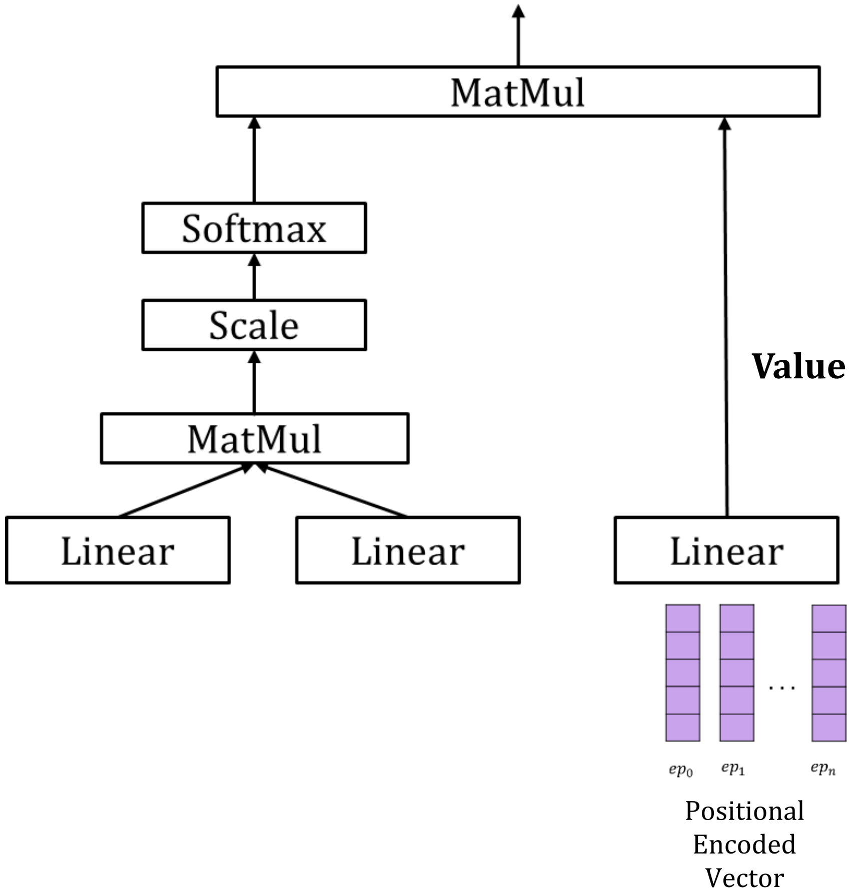




https://medium.com/@hari4om/word-embedding-d816f643140

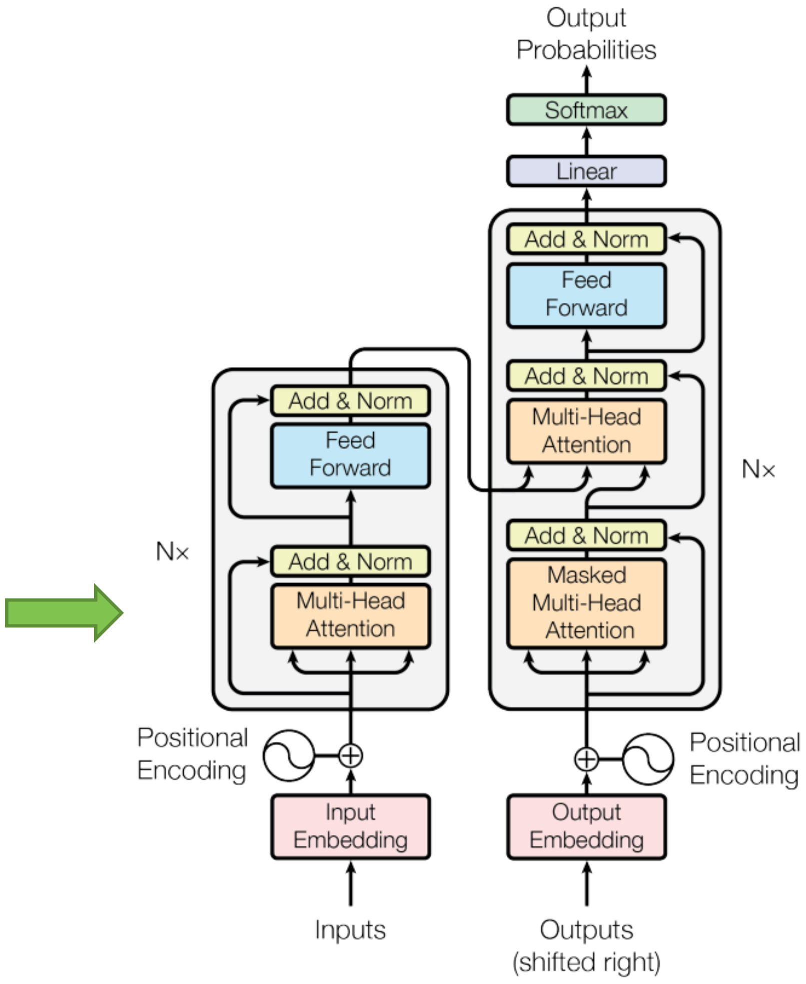
Multi-head Attention

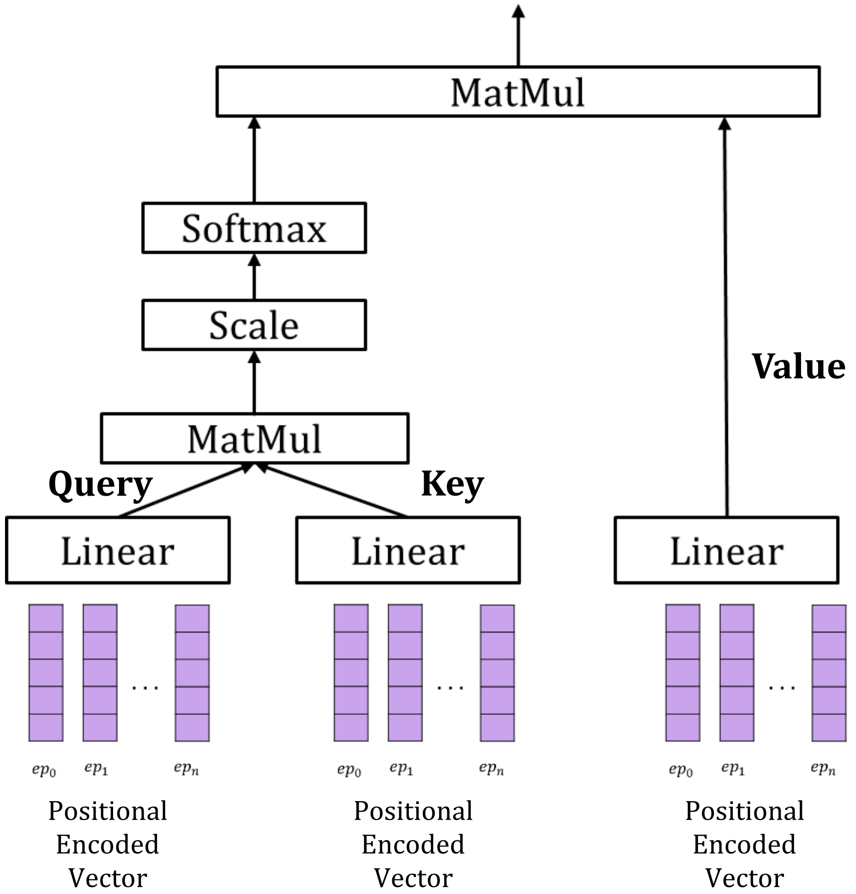




https://medium.com/@hari4om/word-embedding-d816f643140

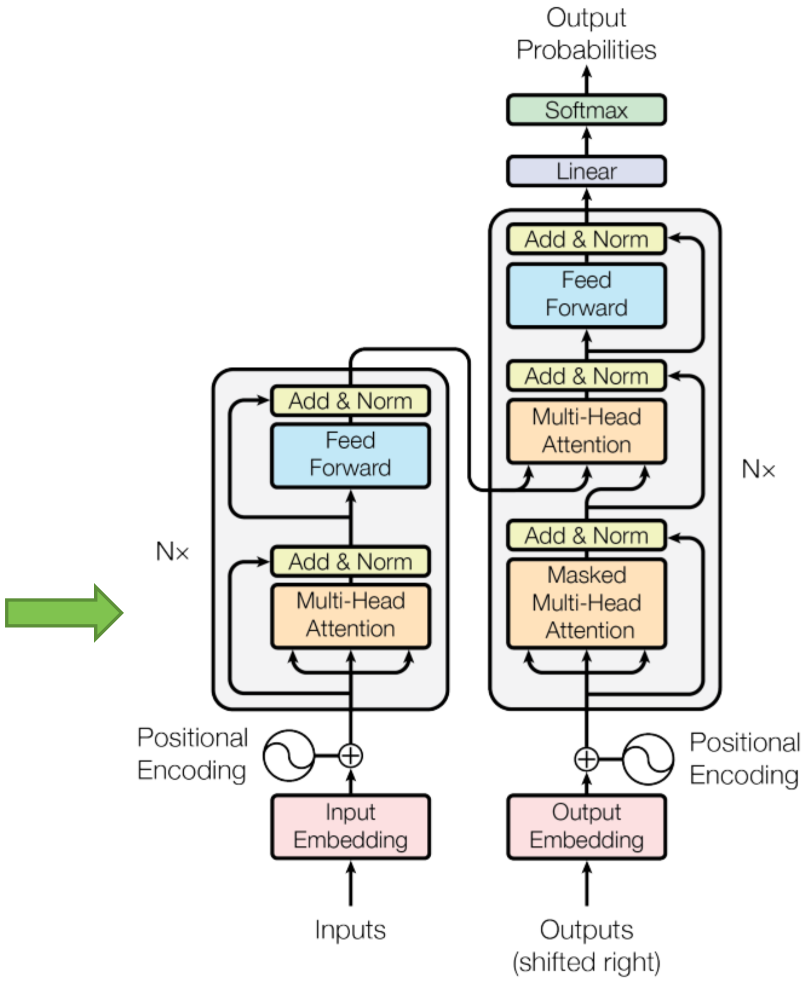
Multi-head Attention



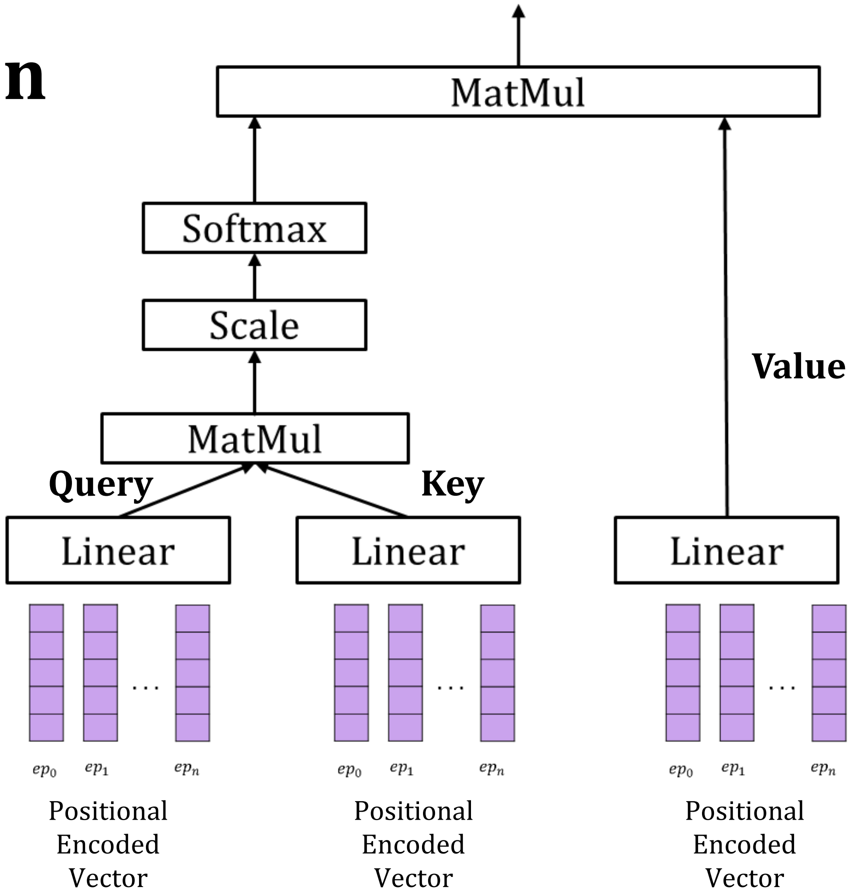


https://medium.com/@hari4om/word-embedding-d816f643140

Multi-head Attention

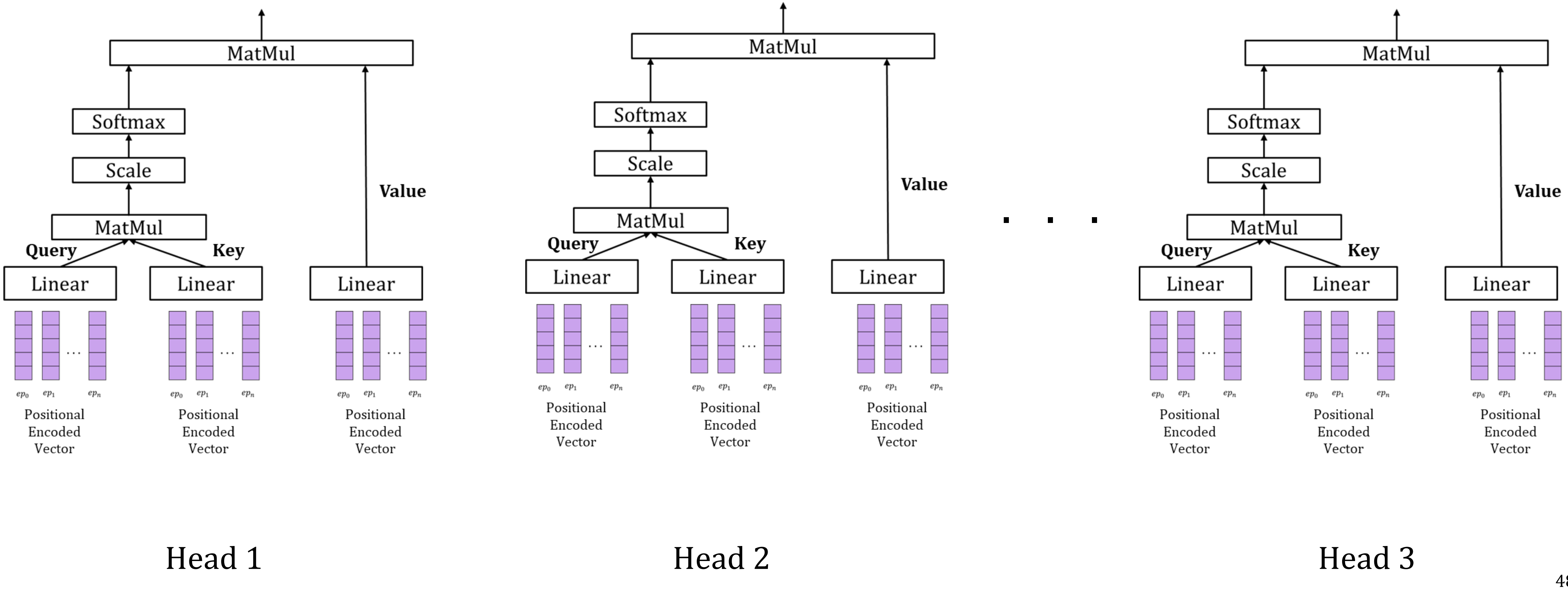


Self-Attention

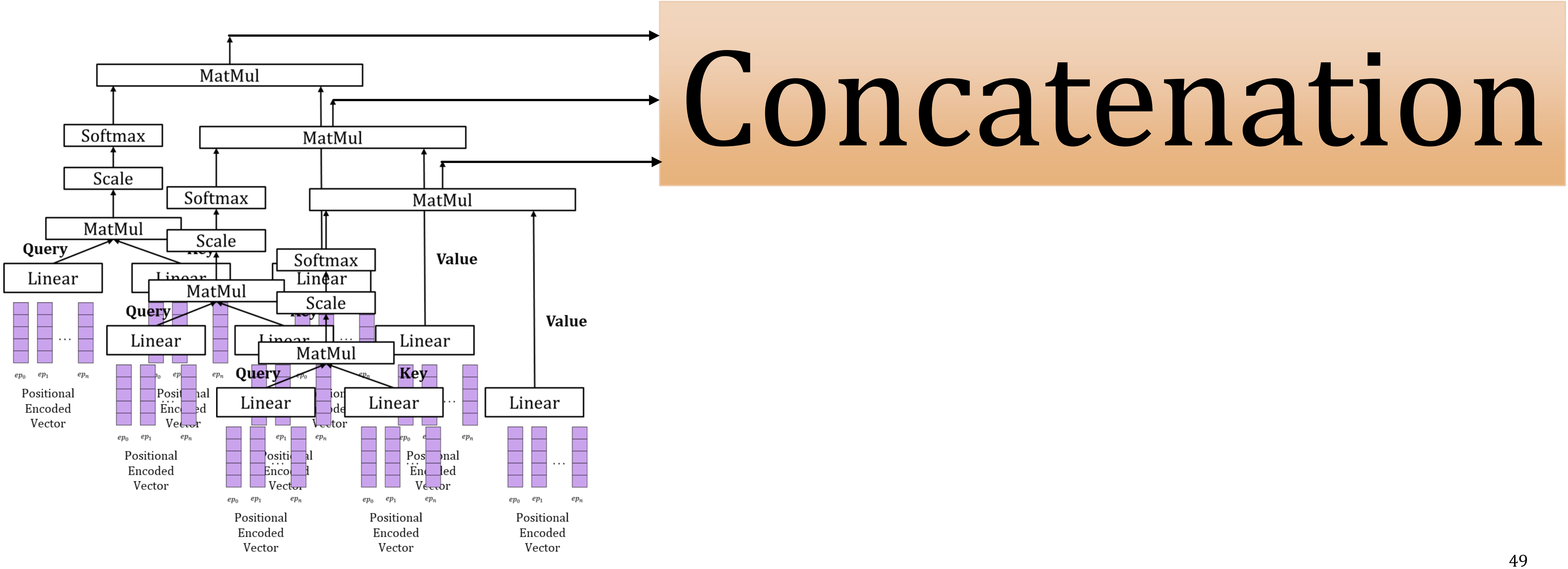


https://medium.com/@hari4om/word-embedding-d816f643140

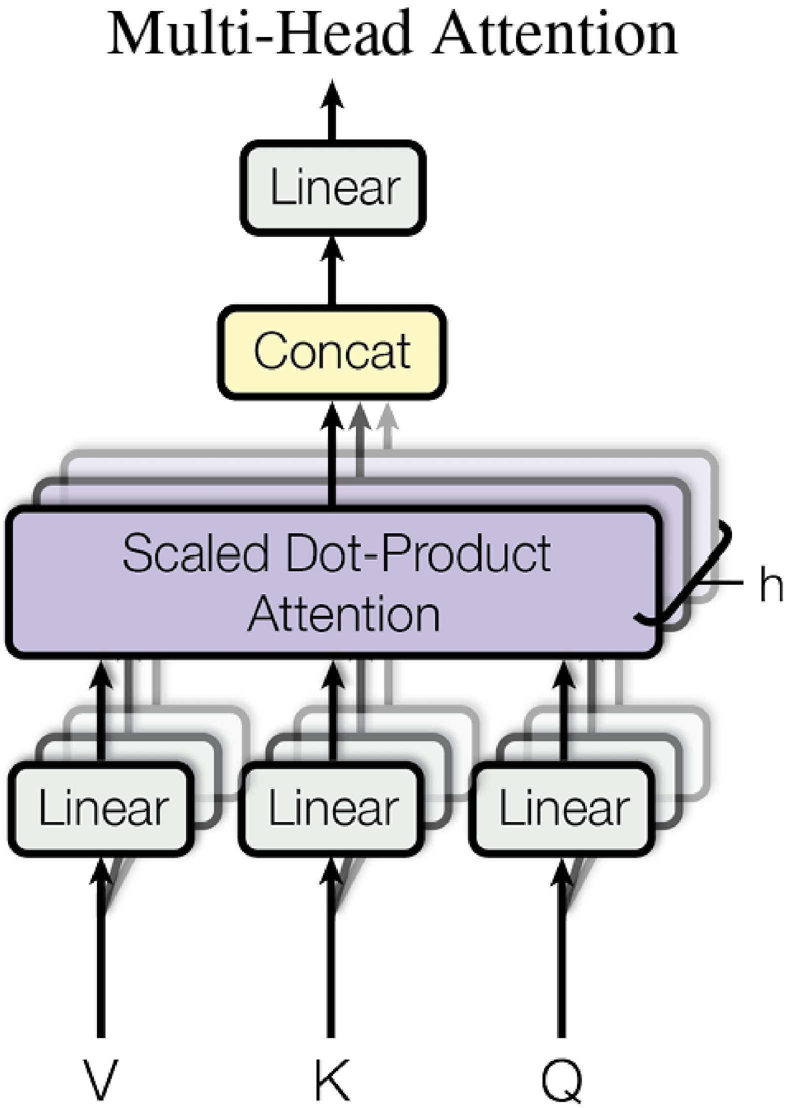
Multi-head Self-Attention



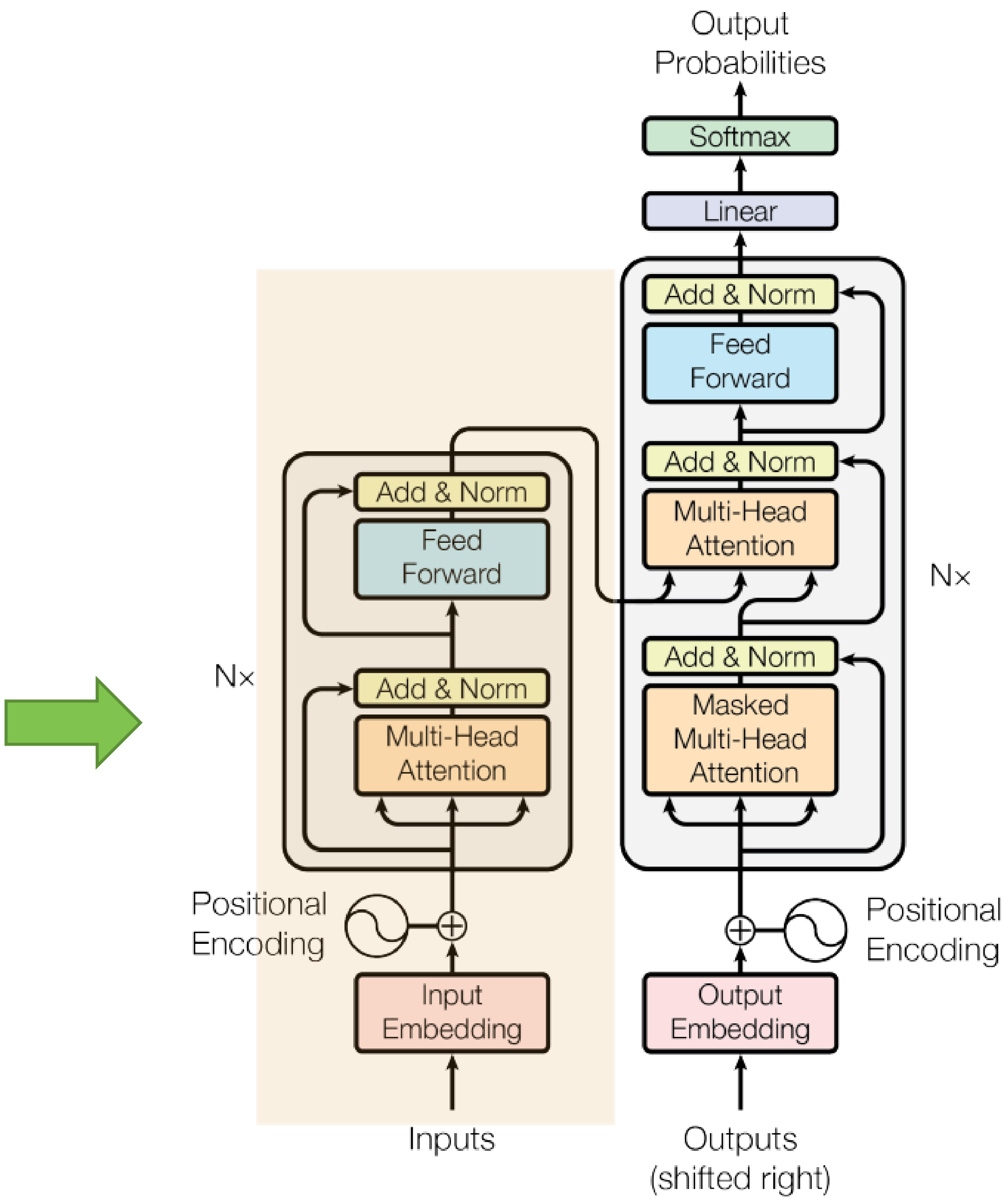
Multi-head Self-Attention



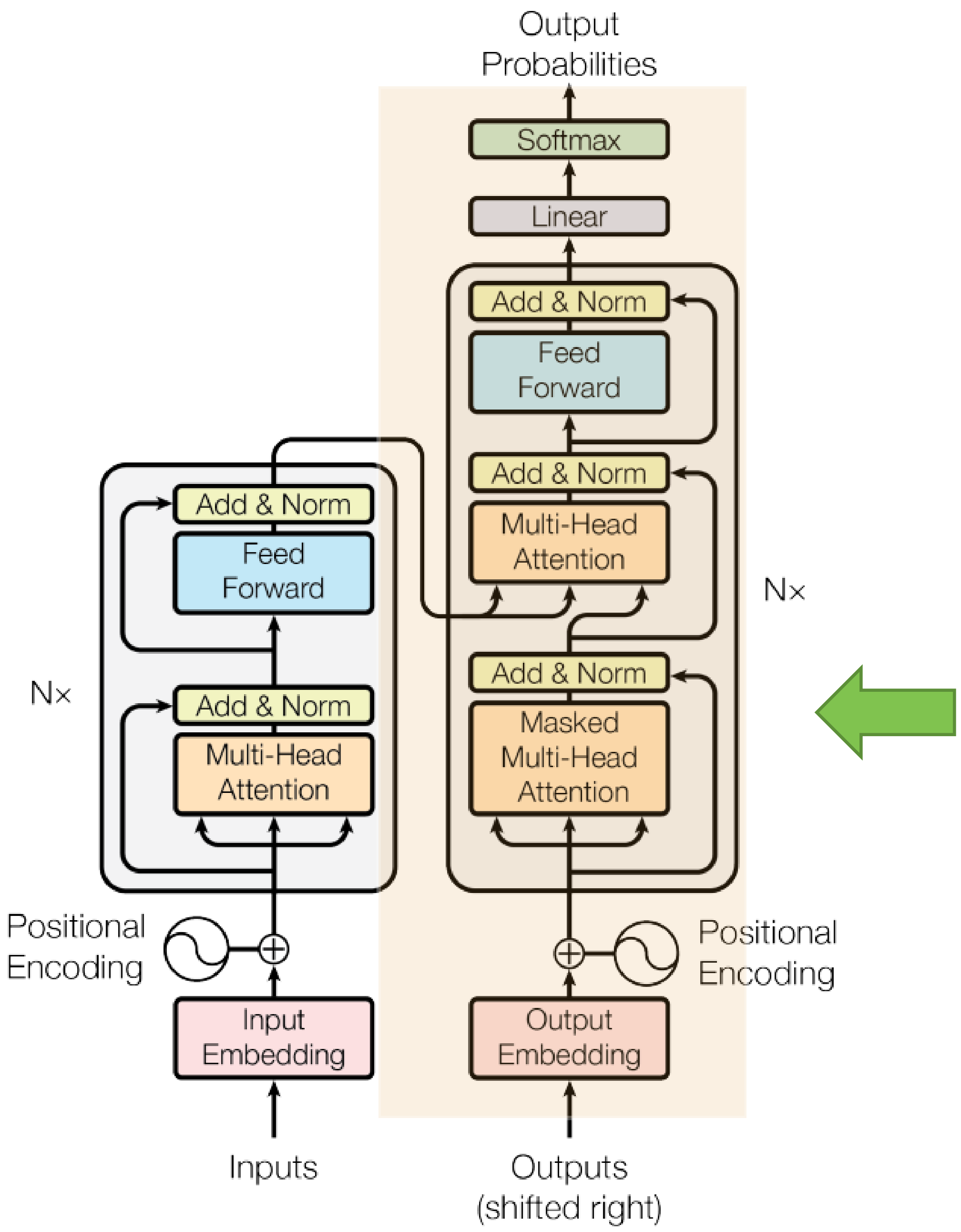
Multi-head Attention



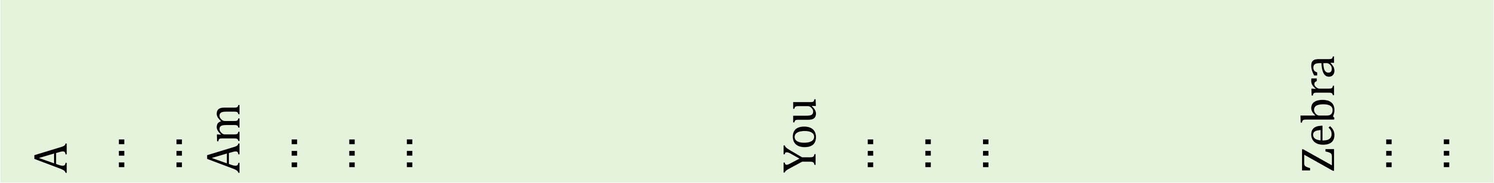
Encoder


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)

Decoder


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)

Transformer: Step by Step



The English vocabulary

(Let’s say ~$50K$ words)

Transformer: Step by Step

A ⋮ ⋮ Am ⋮ ⋮ ⋮ You ⋮ ⋮ ⋮ Zebra ⋮ ⋮

$$\begin{bmatrix}0.7\\0.9\\\cdots\\\cdots\\\cdots\\0.24\end{bmatrix} \cdots\cdots \begin{bmatrix}0.12\\0.35\\\cdots\\\cdots\\\cdots\\0.4\end{bmatrix} \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \begin{bmatrix}0.45\\0.8\\\cdots\\\cdots\\\cdots\\0.34\end{bmatrix}.$$

The English vocabulary (Let's say $\sim 50K$ words)

Each word converted to vector (Word embedding)

Learnt from data

Transformer: Step by Step

s: Embedding dimension

The English vocabulary (Let's say $\sim 50K$ words)

Each word converted to vector (Word embedding)

Learnt from data

In GPT 3, the vocabulary has 50257 tokens

Embedding dimension: 12288

Transformer: Step by Step

Consider the sentence translation problem. We want to translate the following sentence to Hindi

Where there is a will, there is a way

Transformer: Step by Step

Consider the sentence translation problem. We want to translate the following sentence to Hindi

Where there is a will, there is a way

Assume that the Transformer can take at most 20 token-long sentences as input.

Transformer: Step by Step

Where there is a will, there is a way

Assume that the Transformer can take at most 20 token-long sentences as input.

We first divide our input sentence into tokens and pad suitable number of additional tokens so that the total number of tokens become 20

Transformer: Step by Step

Where there is a will, there is a way

Assume that the Transformer can take at most 20 token-long sentences as input.

We first divide our input sentence into tokens and pad suitable number of additional tokens so that the total number of tokens become 20

Where there is a will , there is a way

Transformer: Step by Step

Where there is a will, there is a way

Assume that the Transformer can take at most 20 token-long sentences as input.

We first divide our input sentence into tokens and pad suitable number of additional tokens so that the total number of tokens become 20

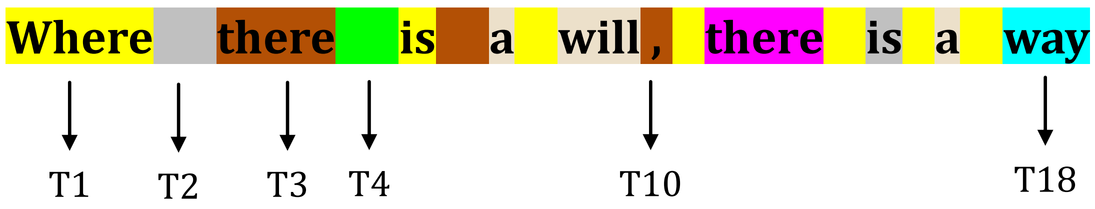
Where there is a will , there is a way

Transformer: Step by Step

Where there is a will, there is a way

Assume that the Transformer can take at most 20 token-long sentences as input.

We first divide our input sentence into tokens and pad suitable number of additional tokens so that the total number of tokens become 20



Transformer: Step by Step

Where there is a will, there is a way

Assume that the Transformer can take at most 20 token-long sentences as input.

We first divide our input sentence into tokens and pad suitable number of additional tokens so that the total number of tokens become 20

Where there is a will , there is a way

T1 T2 T3 T4 T10 T18 T19 T20

Padding tokens

Transformer: Step by Step

Where there is a will, there is a way

T1 T2 T3 T4 T10 T18 Tc

$$\begin{bmatrix} e_{11} \\ e_{12} \\ \cdots \\ \cdots \\ \cdots \\ e_{1s} \end{bmatrix} \quad \begin{bmatrix} e_{21} \\ e_{22} \\ \cdots \\ \cdots \\ \cdots \\ e_{2s} \end{bmatrix} \quad \cdots \quad \begin{bmatrix} e_{c1} \\ e_{c2} \\ \cdots \\ \cdots \\ \cdots \\ e_{cs} \end{bmatrix}$$

c is the max number of tokens

Transformer: Step by Step

Where there is a will, there is a way

T1 T2 T3 T4 T10 T18 Tc

$$\begin{bmatrix} e_{11} \\ e_{12} \\ \dots \\ \dots \\ \dots \\ e_{1s} \end{bmatrix} \quad \begin{bmatrix} e_{21} \\ e_{22} \\ \dots \\ \dots \\ \dots \\ e_{2s} \end{bmatrix} \quad \begin{bmatrix} e_{c1} \\ e_{c2} \\ \dots \\ \dots \\ \dots \\ e_{cs} \end{bmatrix}$$

$$+ \quad + \quad +$$

$$\begin{bmatrix} p_{11} \\ p_{12} \\ \dots \\ \dots \\ \dots \\ p_{1s} \end{bmatrix} \quad \begin{bmatrix} p_{21} \\ p_{22} \\ \dots \\ \dots \\ \dots \\ p_{2s} \end{bmatrix} \quad \begin{bmatrix} p_{c1} \\ p_{c2} \\ \dots \\ \dots \\ \dots \\ p_{cs} \end{bmatrix}$$

Add positional embedding

Transformer: Step by Step

Where there is a will , there is a way

T1 T2 T3 T4 T10 T18 Tc

$$\begin{bmatrix} e_{11} \\ e_{12} \\ \cdots \\ \cdots \\ \cdots \\ e_{1s} \end{bmatrix} + \begin{bmatrix} p_{11} \\ p_{12} \\ \cdots \\ \cdots \\ \cdots \\ p_{1s} \end{bmatrix} = \begin{bmatrix} ep_{11} \\ ep_{12} \\ \cdots \\ \cdots \\ \cdots \\ ep_{1s} \end{bmatrix}$$

$$\begin{bmatrix} e_{21} \\ e_{22} \\ \cdots \\ \cdots \\ \cdots \\ e_{2s} \end{bmatrix} + \begin{bmatrix} p_{21} \\ p_{22} \\ \cdots \\ \cdots \\ \cdots \\ p_{2s} \end{bmatrix} = \begin{bmatrix} ep_{21} \\ ep_{22} \\ \cdots \\ \cdots \\ \cdots \\ ep_{2s} \end{bmatrix}$$

$$\begin{bmatrix} e_{c1} \\ e_{c2} \\ \cdots \\ \cdots \\ \cdots \\ e_{cs} \end{bmatrix} + \begin{bmatrix} p_{c1} \\ p_{c2} \\ \cdots \\ \cdots \\ \cdots \\ p_{cs} \end{bmatrix} = \begin{bmatrix} ep_{c1} \\ ep_{c2} \\ \cdots \\ \cdots \\ \cdots \\ ep_{cs} \end{bmatrix}$$

Transformer: Step by Step

Where there is a will, there is a way

T1 T2 T3 T4 T10 T18 Tc

$$\underbrace{\begin{bmatrix} e_{11}\\ e_{12}\\ \cdots\\ \cdots\\ \cdots\\ e_{1s}\end{bmatrix} + \begin{bmatrix} p_{11}\\ p_{12}\\ \cdots\\ \cdots\\ \cdots\\ p_{1s}\end{bmatrix} = \begin{bmatrix} ep_{11}\\ ep_{12}\\ \cdots\\ \cdots\\ \cdots\\ ep_{1s}\end{bmatrix}}_{\boldsymbol{ep_1}}$$

$$\underbrace{\begin{bmatrix} e_{21}\\ e_{22}\\ \cdots\\ \cdots\\ \cdots\\ e_{2s}\end{bmatrix} + \begin{bmatrix} p_{21}\\ p_{22}\\ \cdots\\ \cdots\\ \cdots\\ p_{2s}\end{bmatrix} = \begin{bmatrix} ep_{21}\\ ep_{22}\\ \cdots\\ \cdots\\ \cdots\\ ep_{2s}\end{bmatrix}}_{\boldsymbol{ep_2}}$$

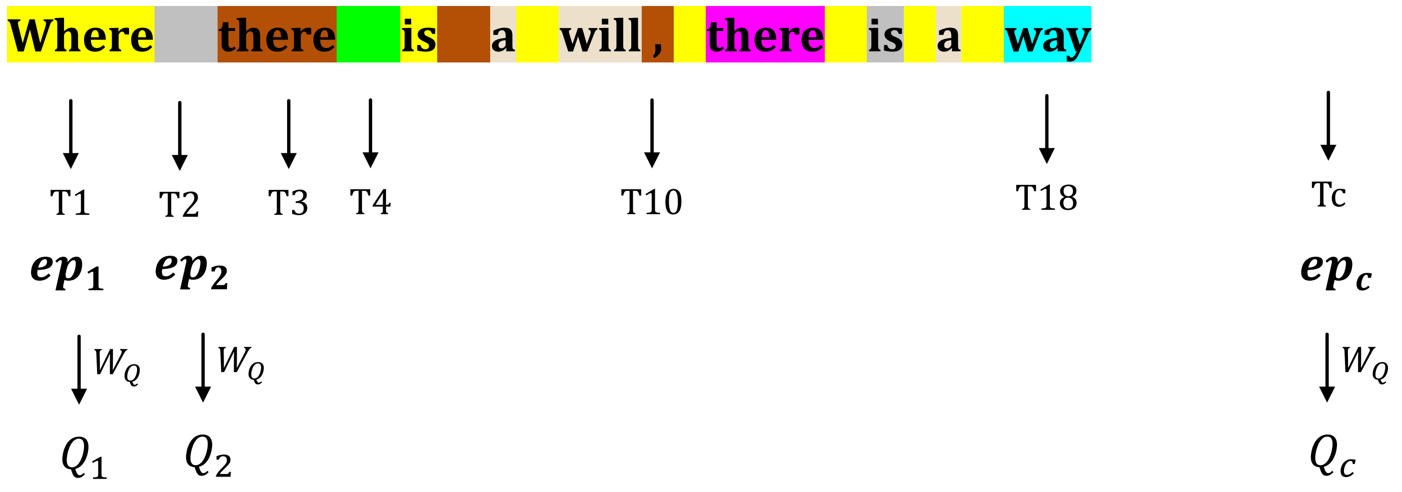
$$\underbrace{\begin{bmatrix} e_{c1}\\ e_{c2}\\ \cdots\\ \cdots\\ \cdots\\ e_{cs}\end{bmatrix} + \begin{bmatrix} p_{c1}\\ p_{c2}\\ \cdots\\ \cdots\\ \cdots\\ p_{cs}\end{bmatrix} = \begin{bmatrix} ep_{c1}\\ ep_{c2}\\ \cdots\\ \cdots\\ \cdots\\ ep_{cs}\end{bmatrix}}_{\boldsymbol{ep_c}}$$

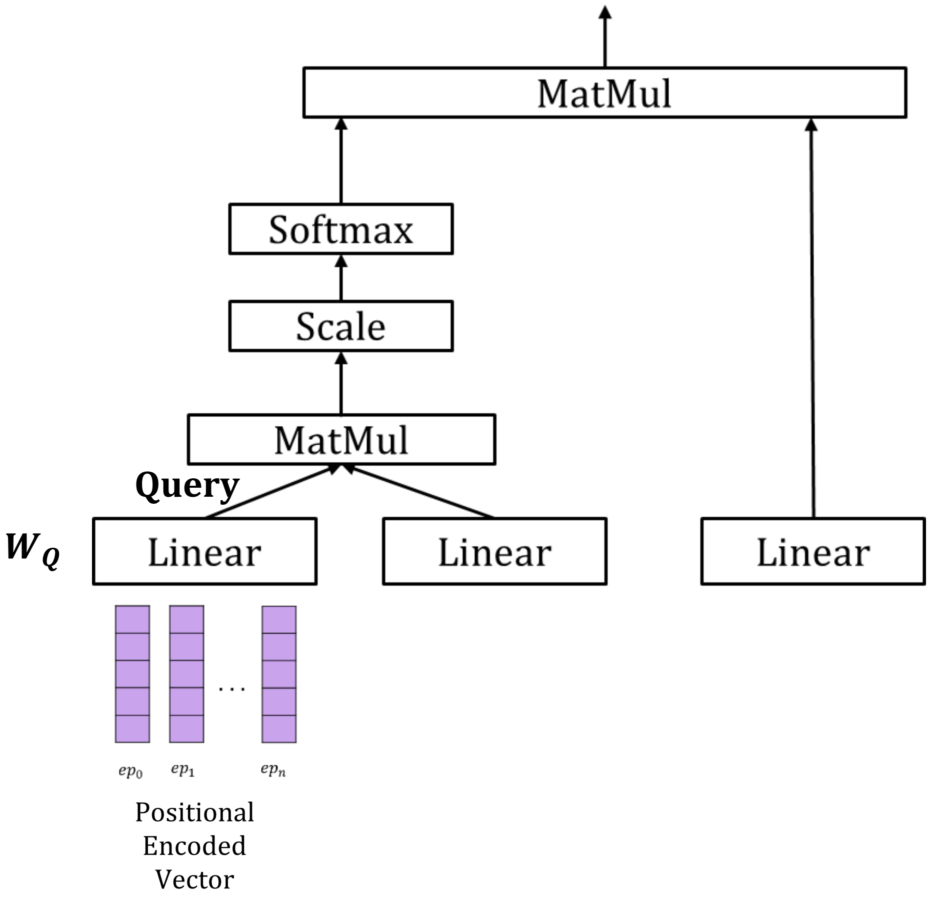
Where there is a will, there is a way

T1 T2 T3 T4 T10 T18 Tc

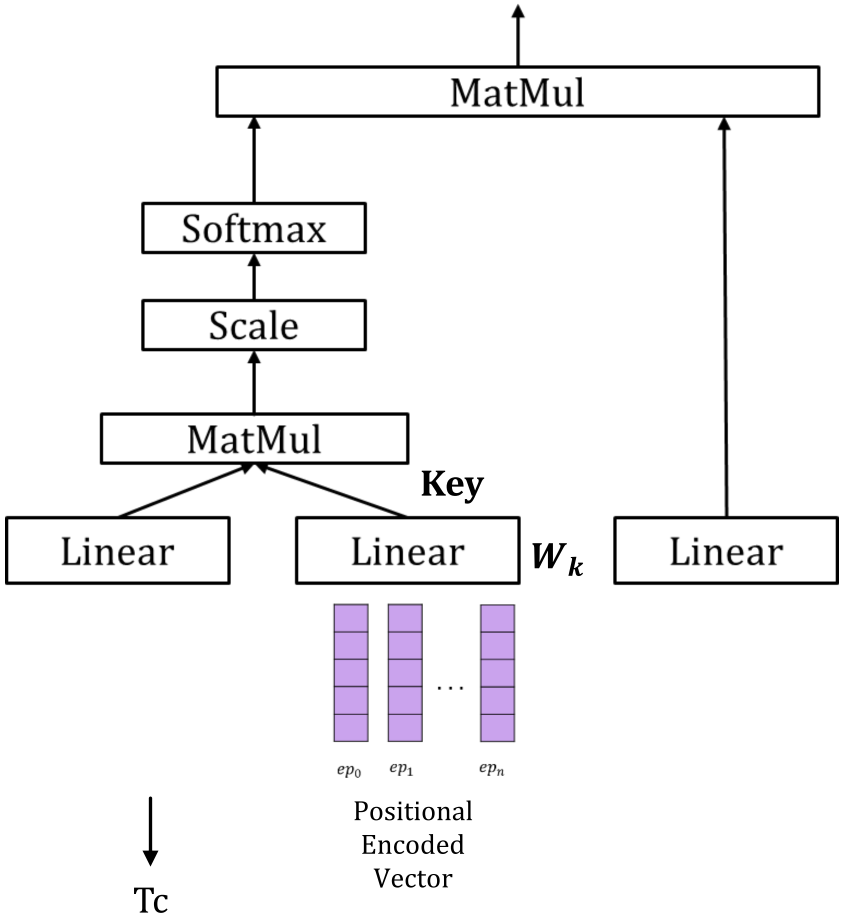
$\boldsymbol{ep_1}$ $\boldsymbol{ep_2}$ $\boldsymbol{ep_c}$

Transformer: Step by Step





Q_i: Query vector of dimension d for i^{th} token



Where there is a will, there is a way

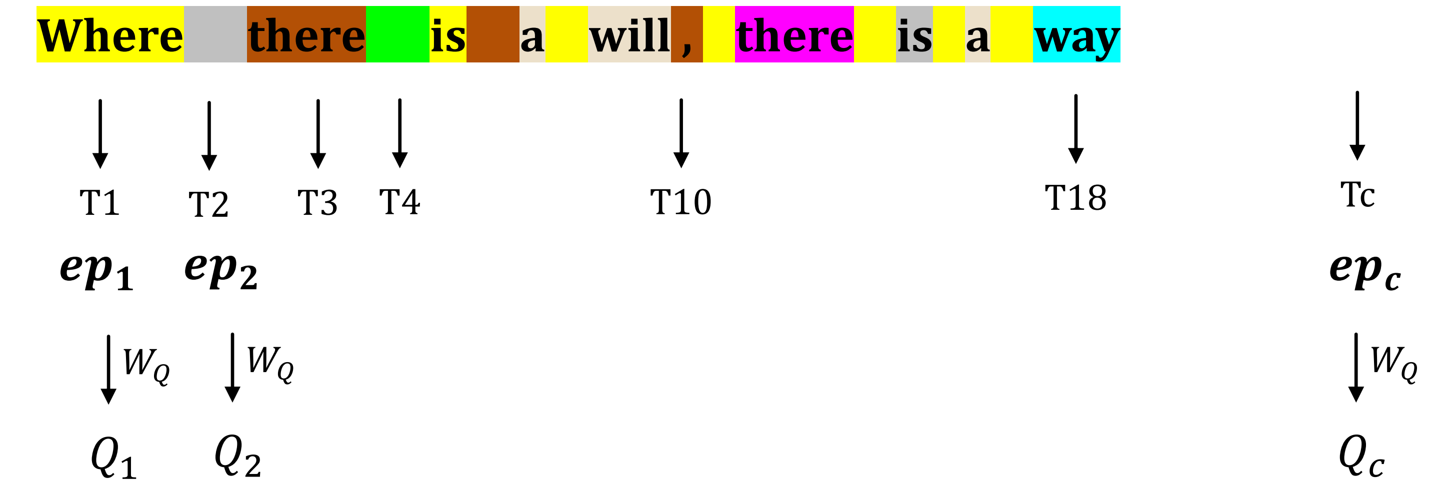
T1 T2 T3 T4 T10 T18 Tc

$\boldsymbol{ep_1} \xrightarrow{W_k} K_1$

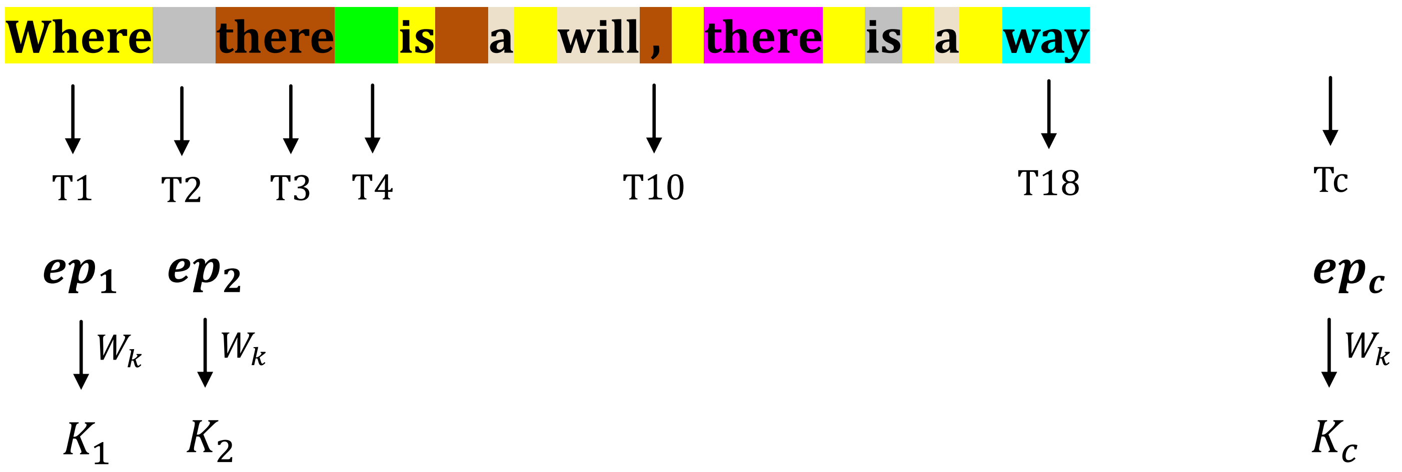
$\boldsymbol{ep_2} \xrightarrow{W_k} K_2$

$\boldsymbol{ep_c} \xrightarrow{W_k} K_c$

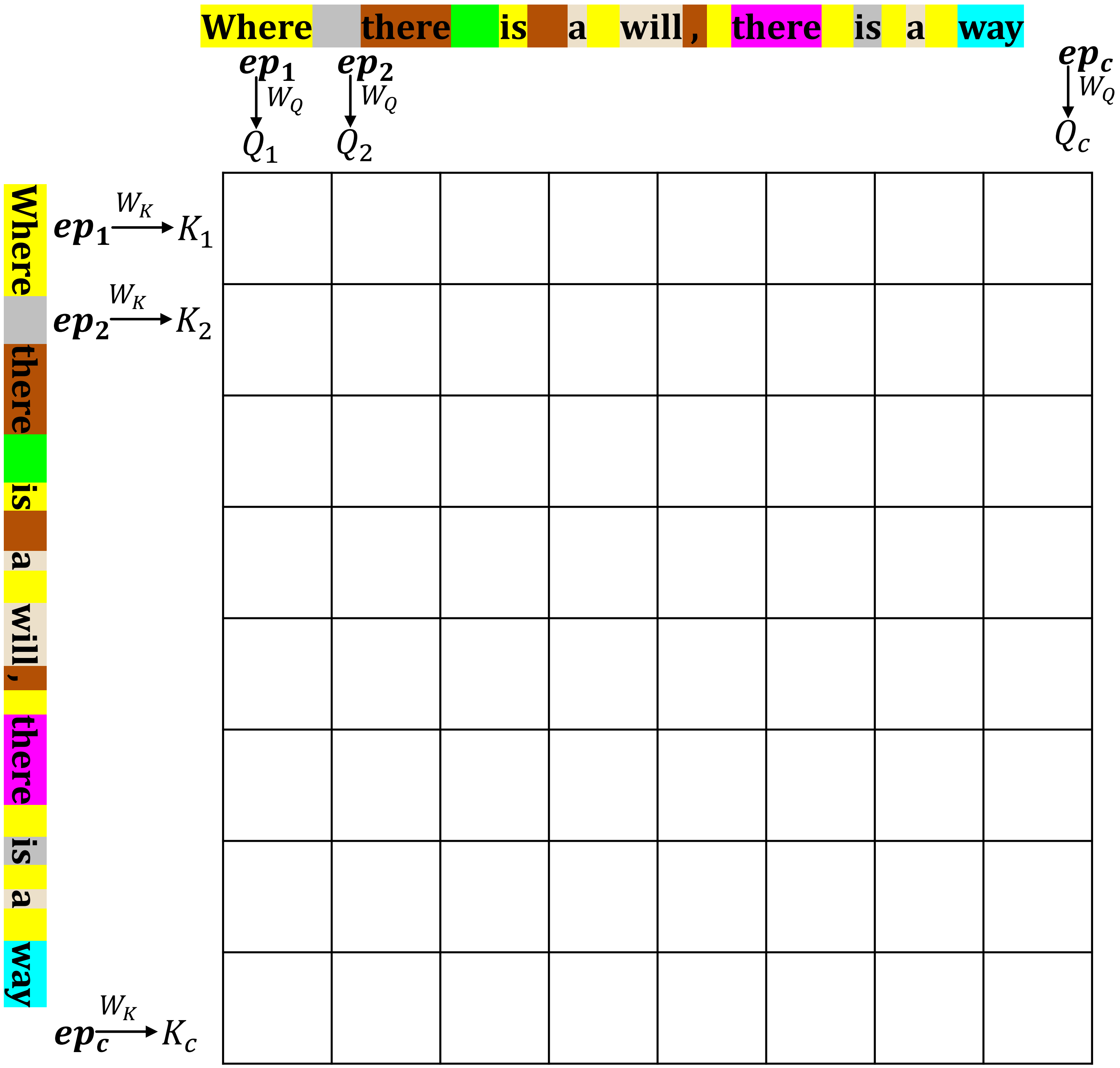
K_i: Key vector of dimension d for i^{th} token



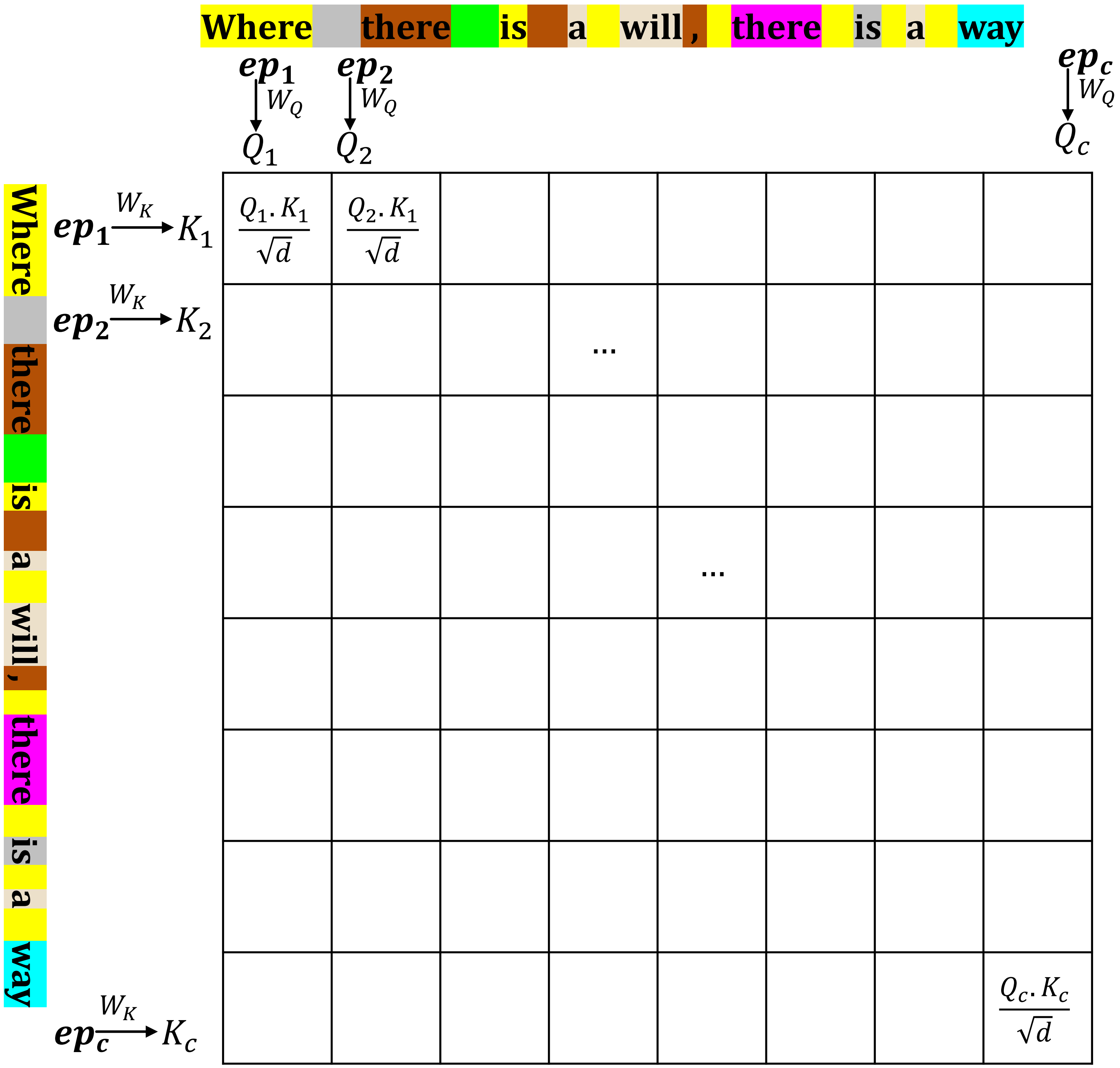
Q_i: Query vector of dimension d for i^{th} token

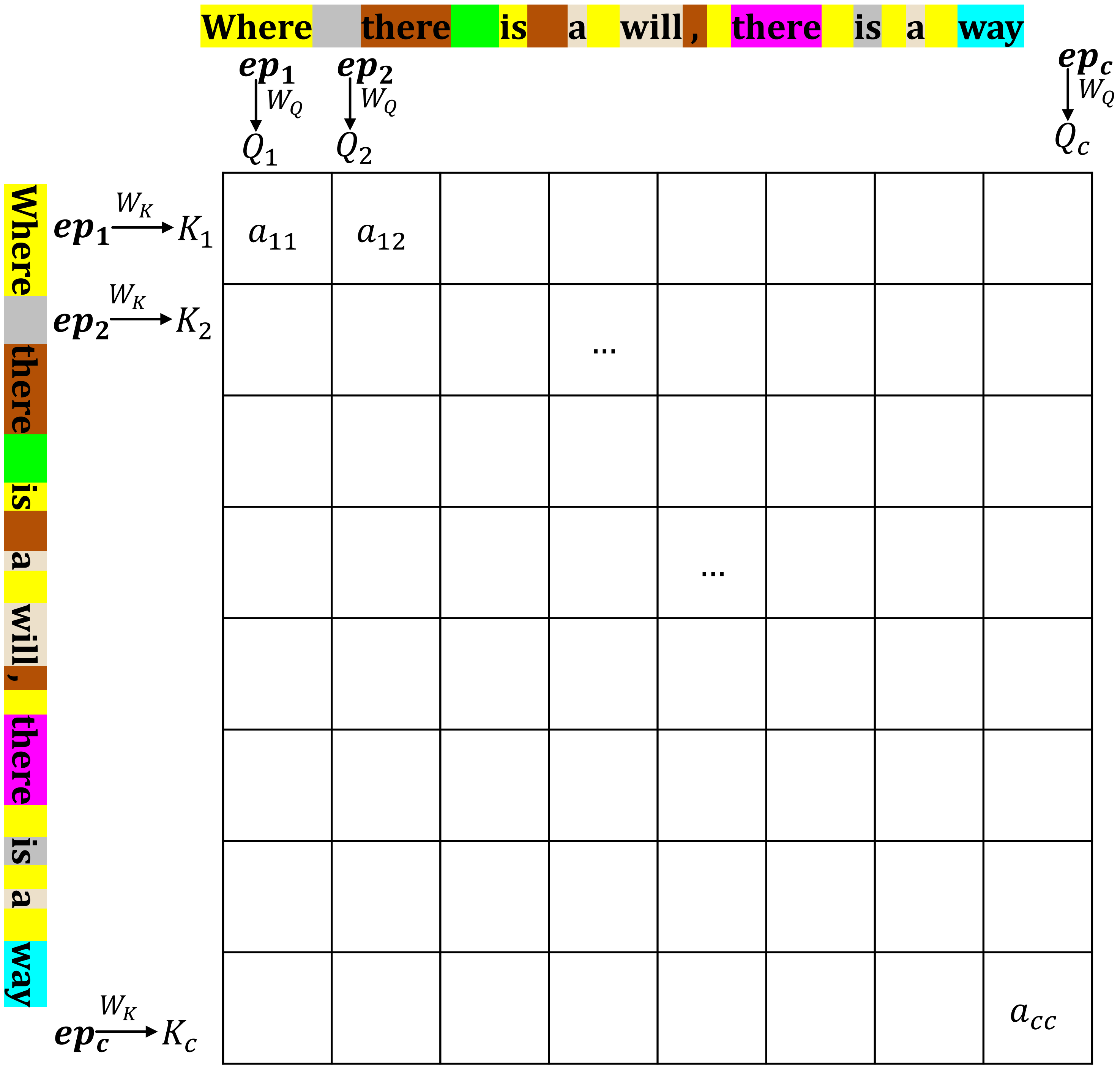


K_i: Key vector of dimension d for i^{th} token



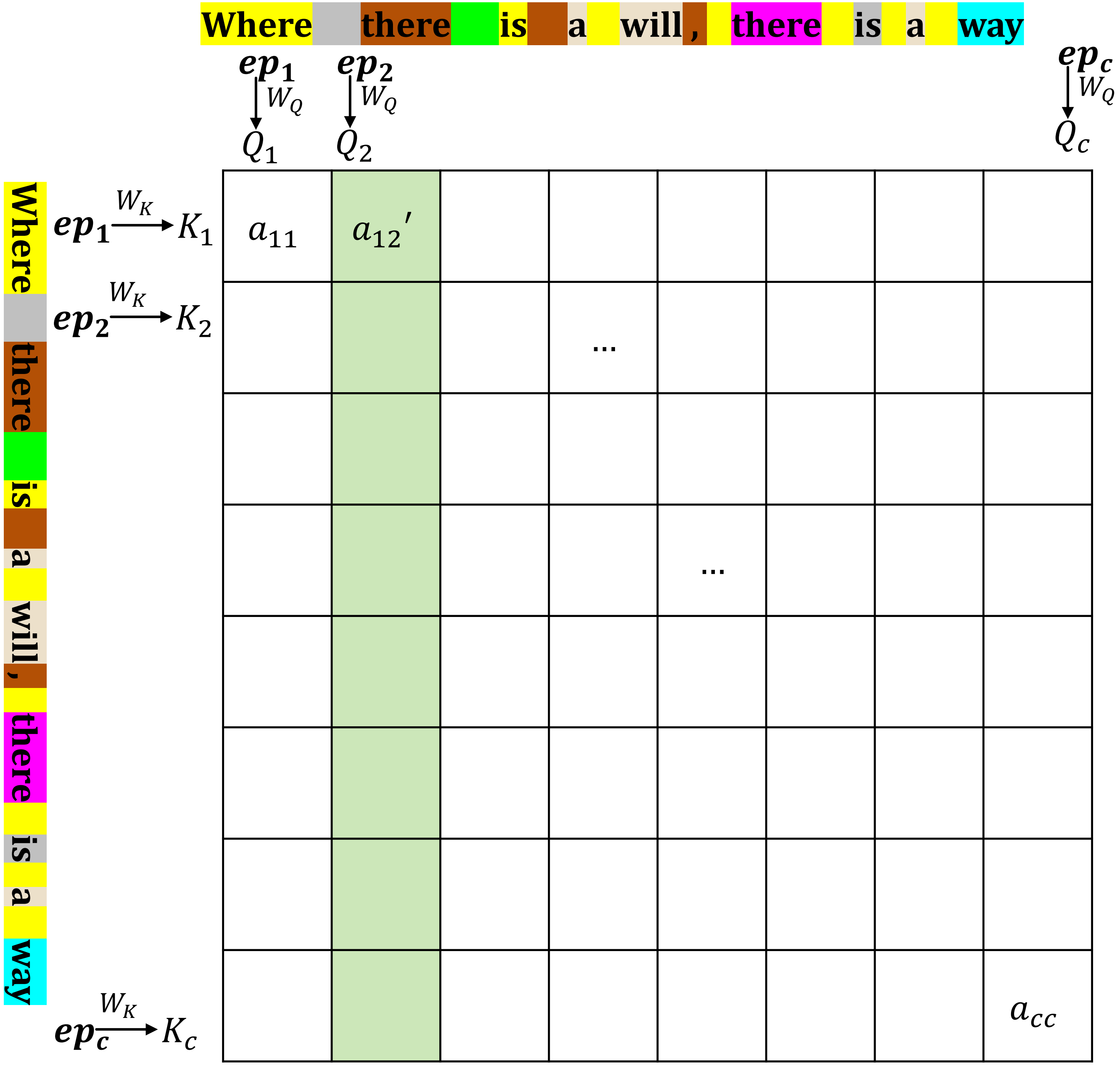
Where there is a will , there is a way
$ep_1 \xrightarrow{W_Q} Q_1$
$ep_2 \xrightarrow{W_Q} Q_2$
$ep_c \xrightarrow{W_Q} Q_c$
Where there is a will , there is a way
$ep_1 \xrightarrow{W_K} K_1$
$ep_2 \xrightarrow{W_K} K_2$
$ep_c \xrightarrow{W_K} K_c$


Where there is a will, there is a way
$ep_1 \xrightarrow{W_Q} Q_1$
$ep_2 \xrightarrow{W_Q} Q_2$
$ep_c \xrightarrow{W_Q} Q_c$
$ep_1 \xrightarrow{W_K} K_1$
$ep_2 \xrightarrow{W_K} K_2$
$ep_c \xrightarrow{W_K} K_c$
$\frac{Q_1.K_1}{\sqrt{d}}$
$\frac{Q_2.K_1}{\sqrt{d}}$
...
...
$\frac{Q_c.K_c}{\sqrt{d}}$



Attention matrix

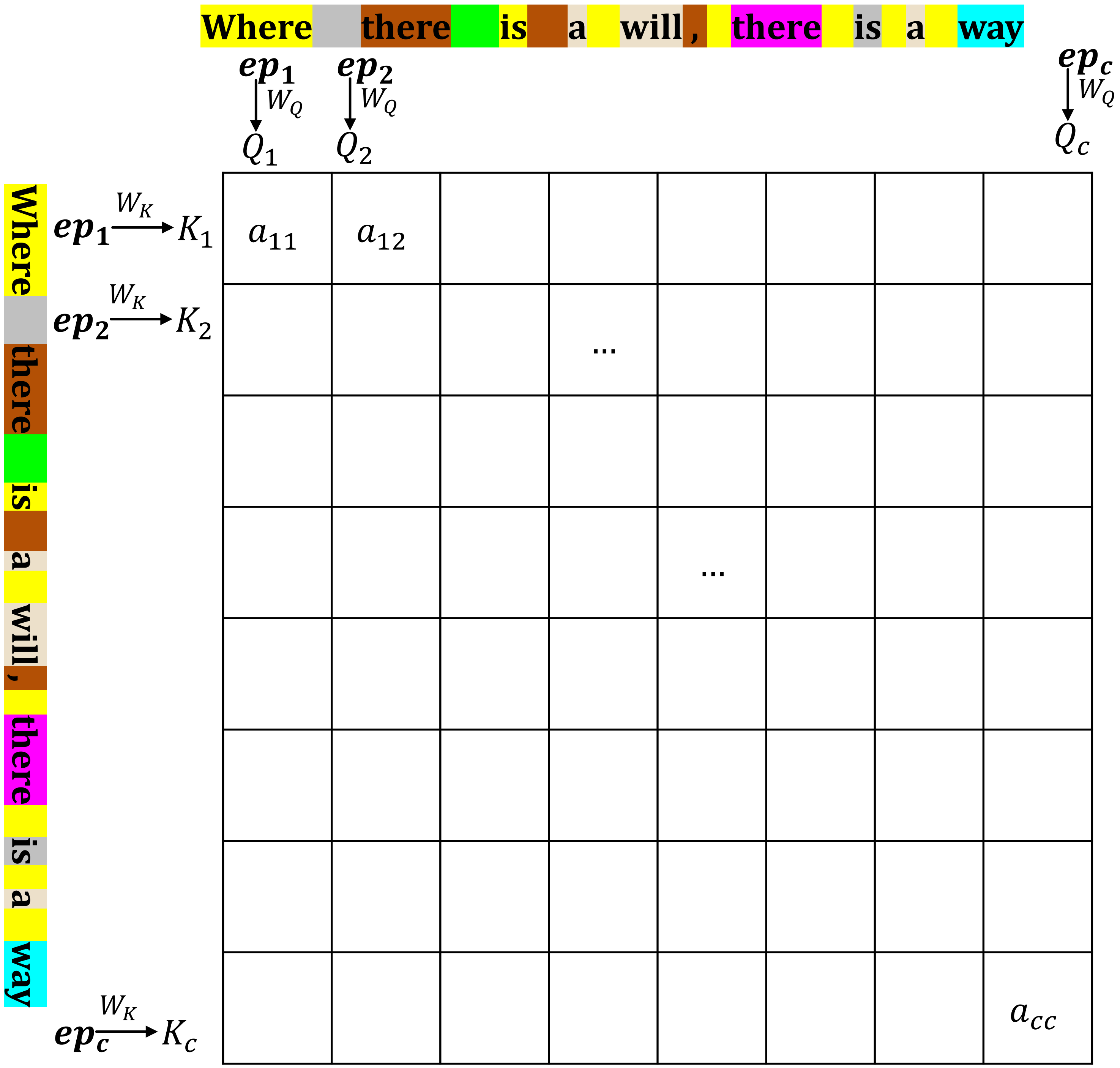
Entry (i, j) indicate some sort of correlation between the token of the i^{th} row and the token of the j^{th} row



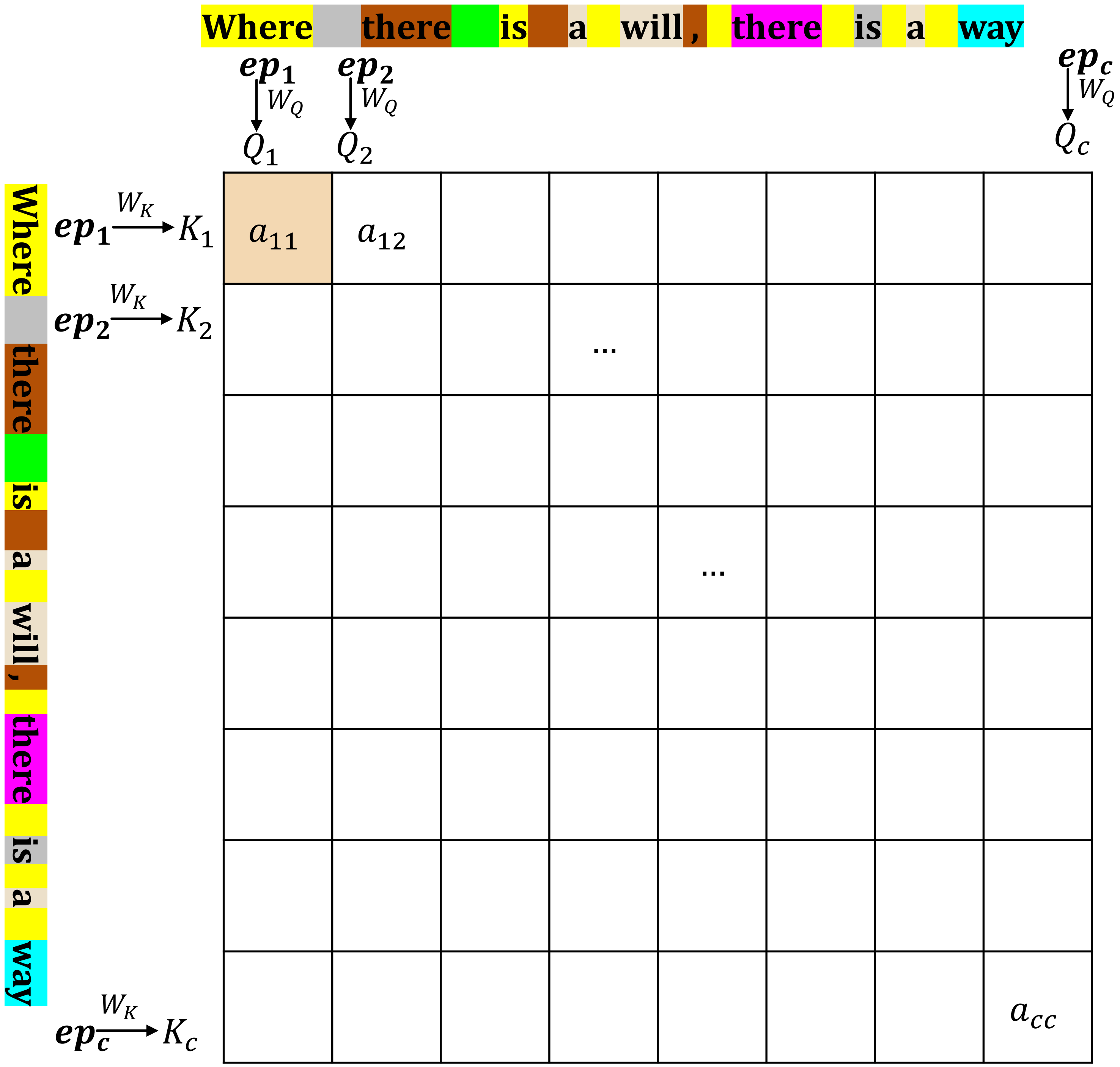
Attention matrix

For each query column, we apply softmax. As a result, each column kind of indicates the probabilities of the corresponding query to match with different keys

a'_{12}: Value of the entry after softmax

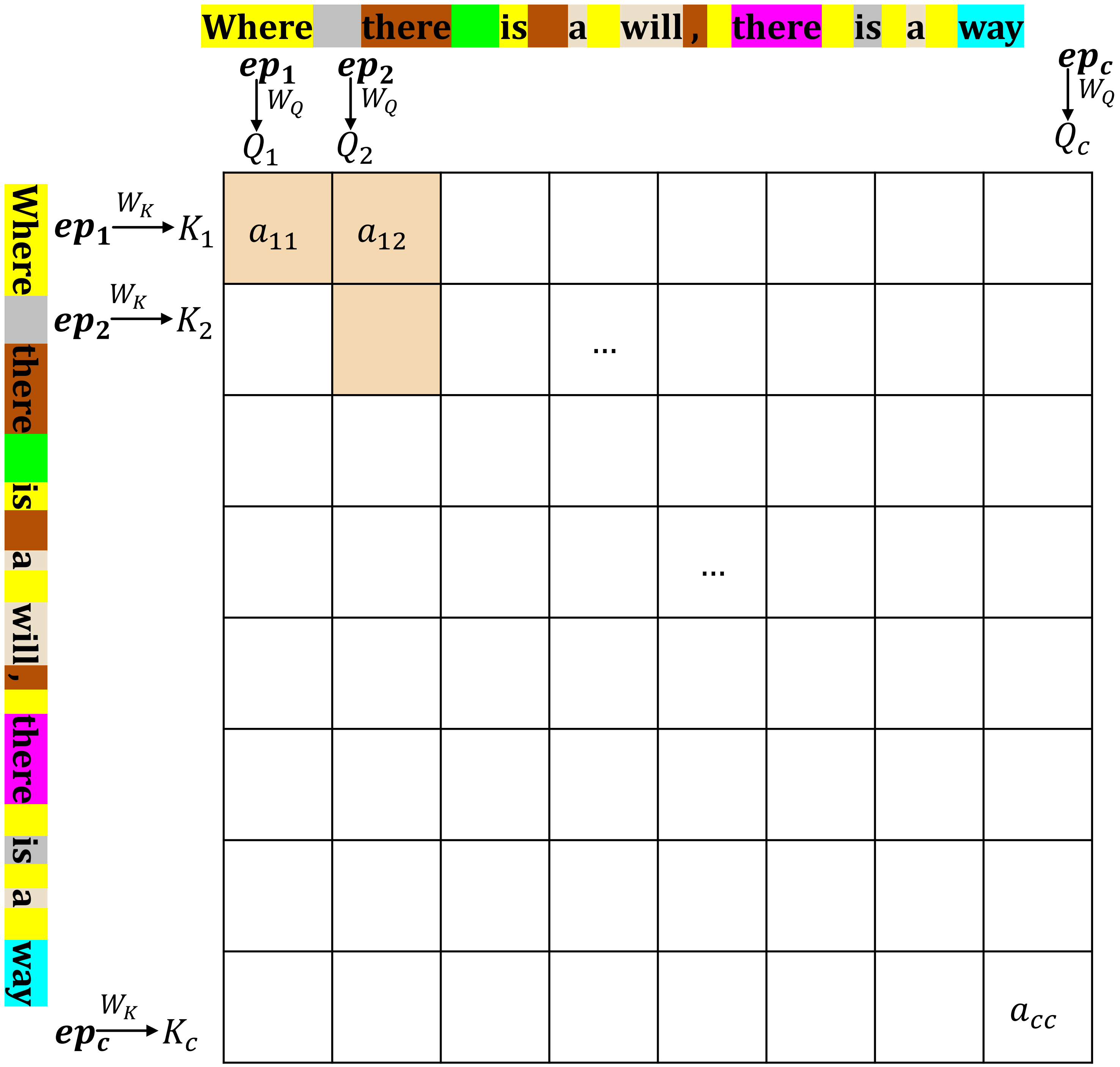


We may want that a query (a token) is influenced by only past keys (past tokens)



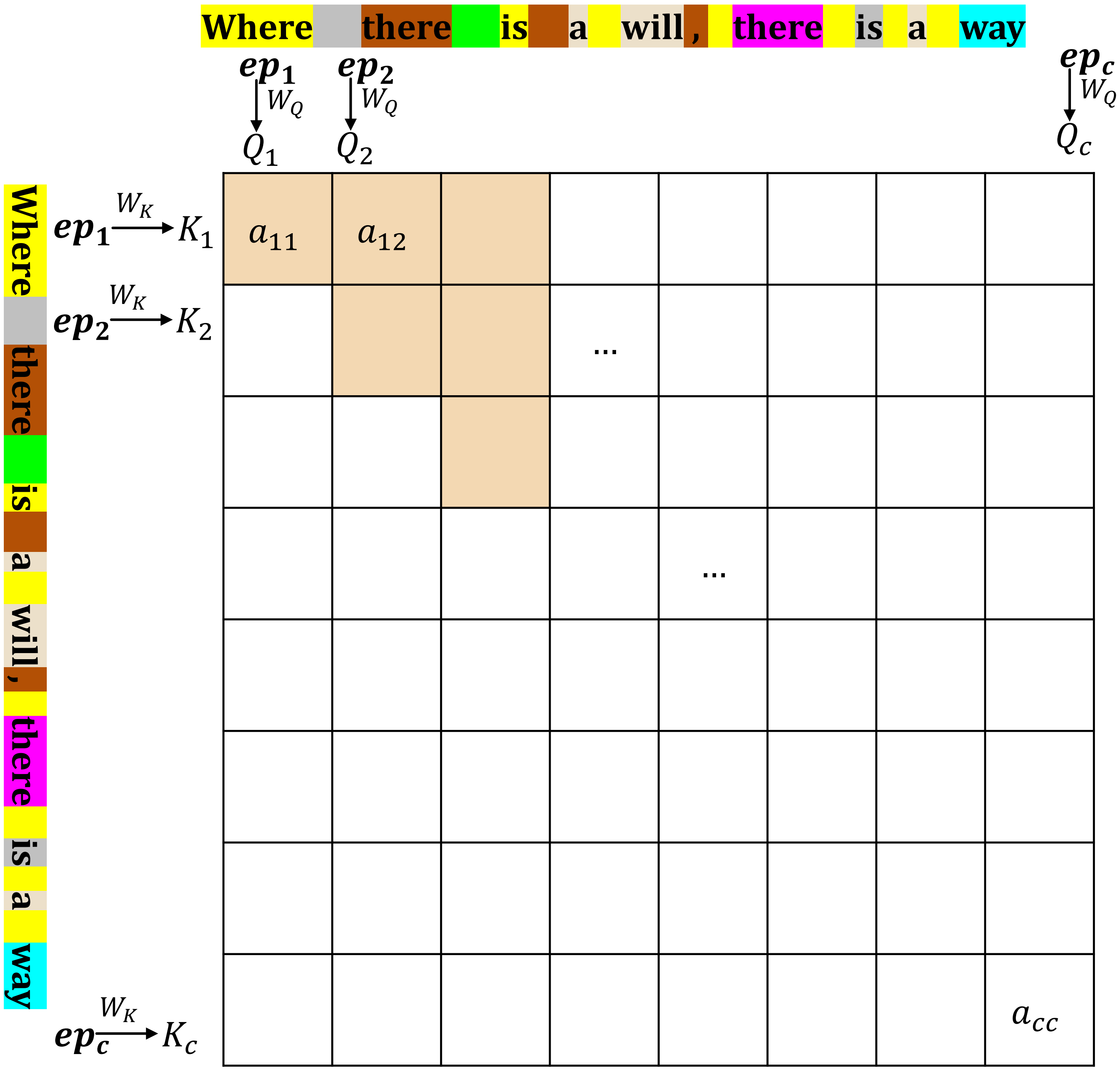
We may want that a query (a token) is influenced by only past keys (past tokens)

Q_1 is influenced by only K_1



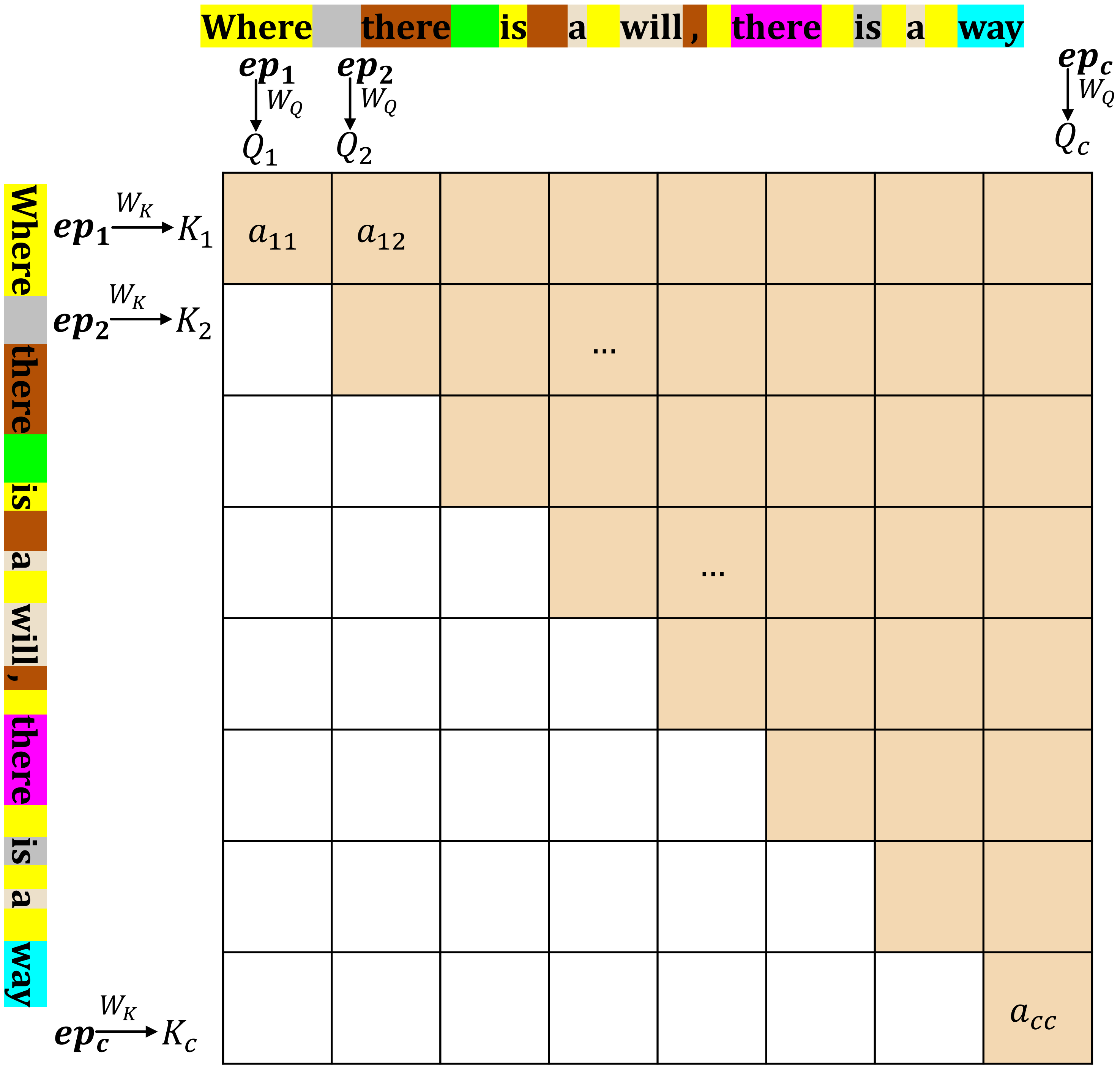
We may want that a query (a token) is influenced by only past keys (past tokens)

Q_1 is influenced by only K_1
Q_2 is influenced by only K_1, K_2



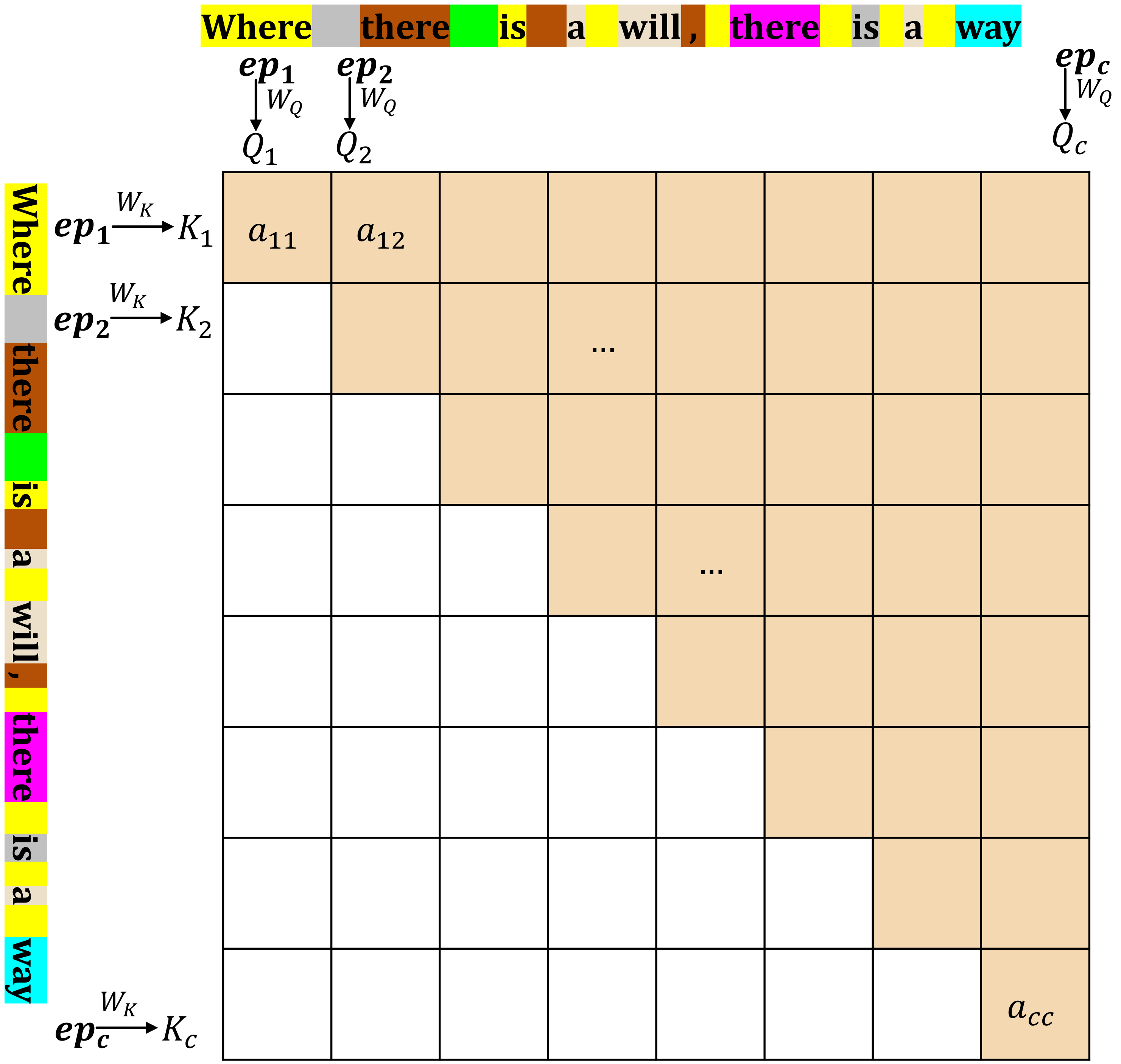
We may want that a query (a token) is influenced by only past keys (past tokens)

Q_1 is influenced by only K_1
Q_2 is influenced by only K_1, K_2
Q_3 is influenced by only K_1, K_2, K_3, and so on



We may want that a query (a token) is influenced by only past keys (past tokens)

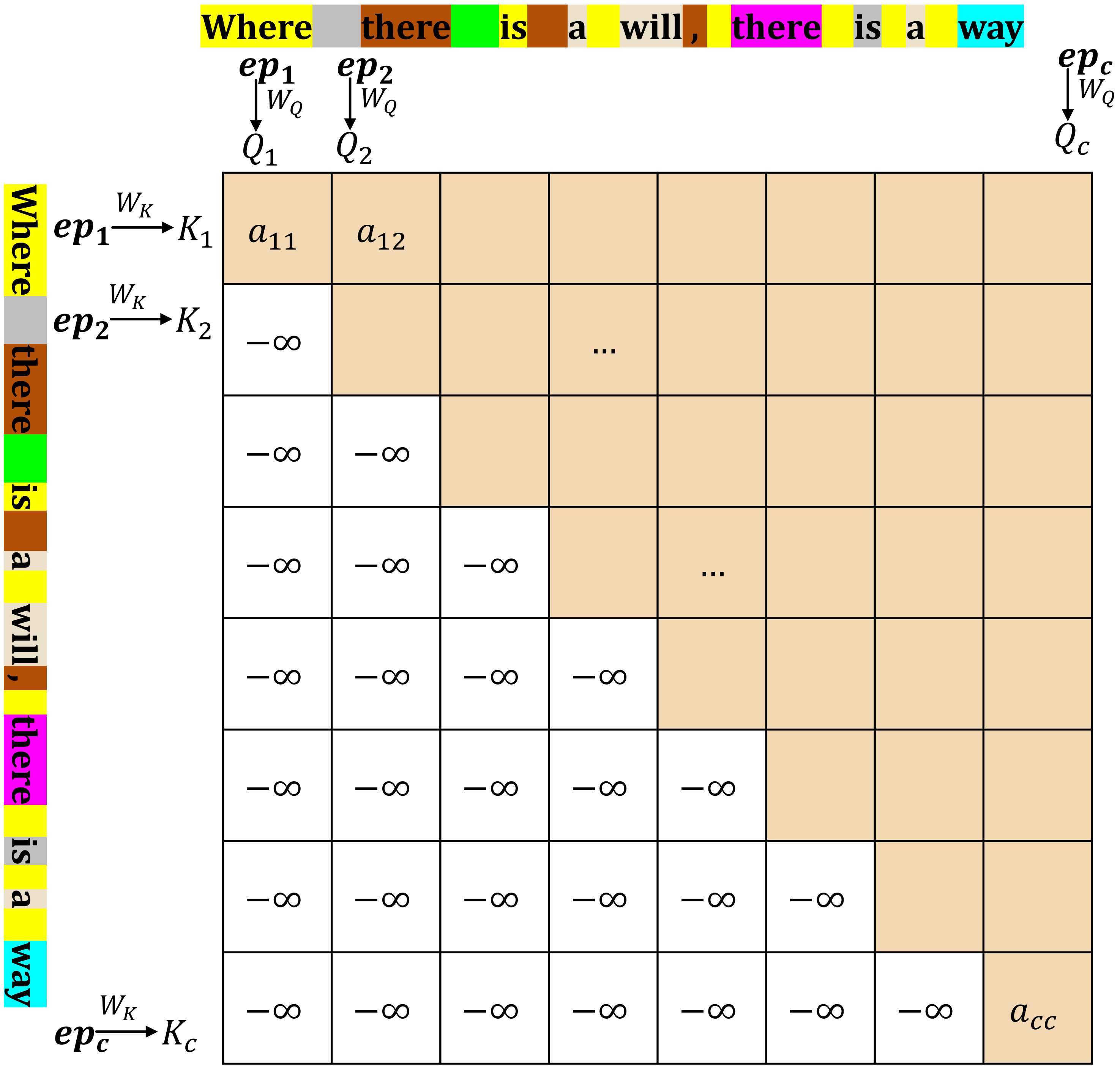
Q_1 is influenced by only K_1
Q_2 is influenced by only K_1, K_2
Q_3 is influenced by only K_1, K_2, K_3, and so on



We may want that a query (a token) is influenced by only past keys (past tokens)

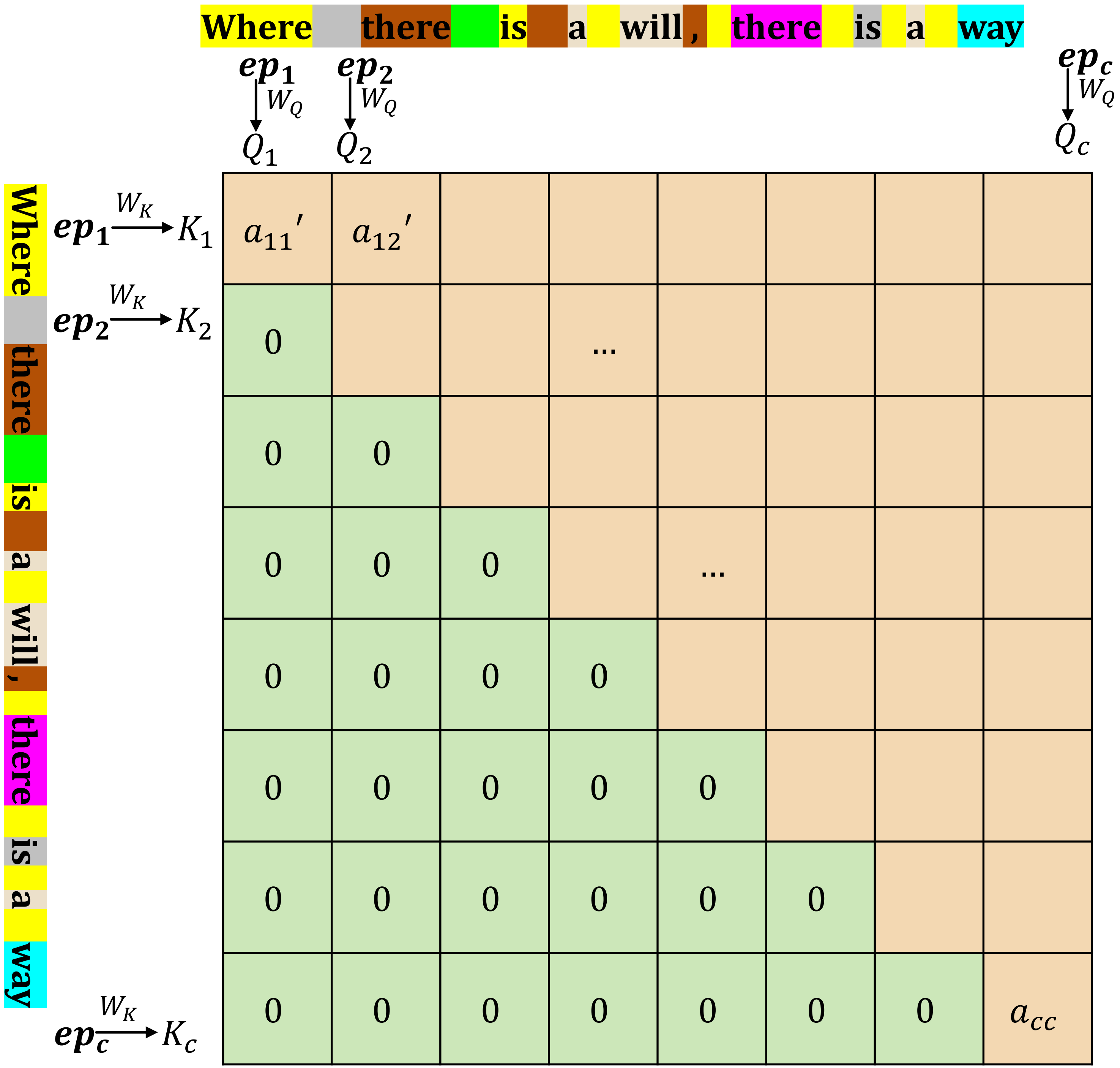
Q_1 is influenced by only K_1
Q_2 is influenced by only K_1, K_2
Q_3 is influenced by only K_1, K_2, K_3, and so on

This is equivalent to masking the white region



This is equivalent to masking the white region

For that, we make the entries at the white boxes $-\infty$

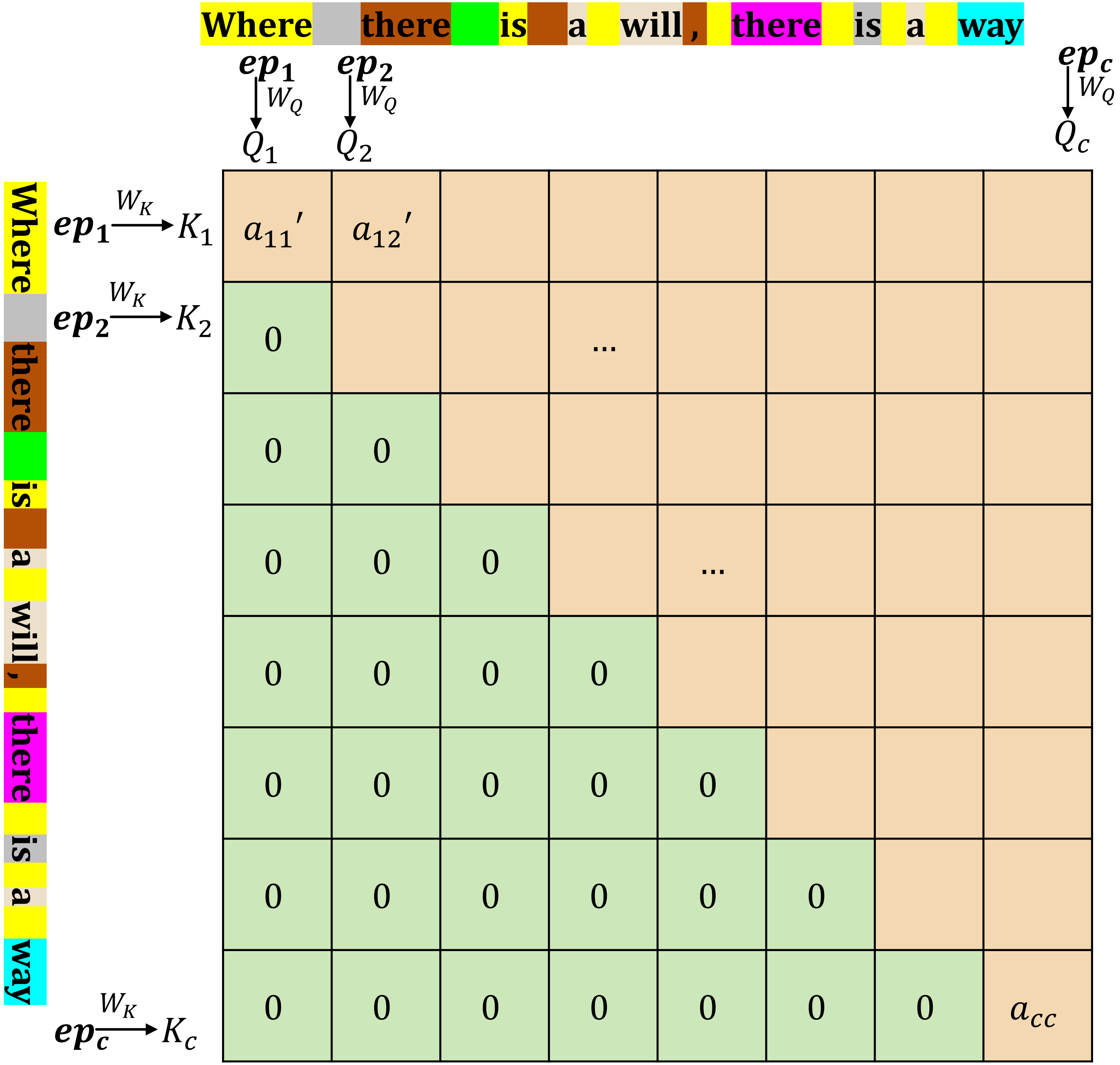


This is equivalent to masking the white region

For that, we make the entries at the white boxes $-\infty$

After column-wise softmax, these entries become 0

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

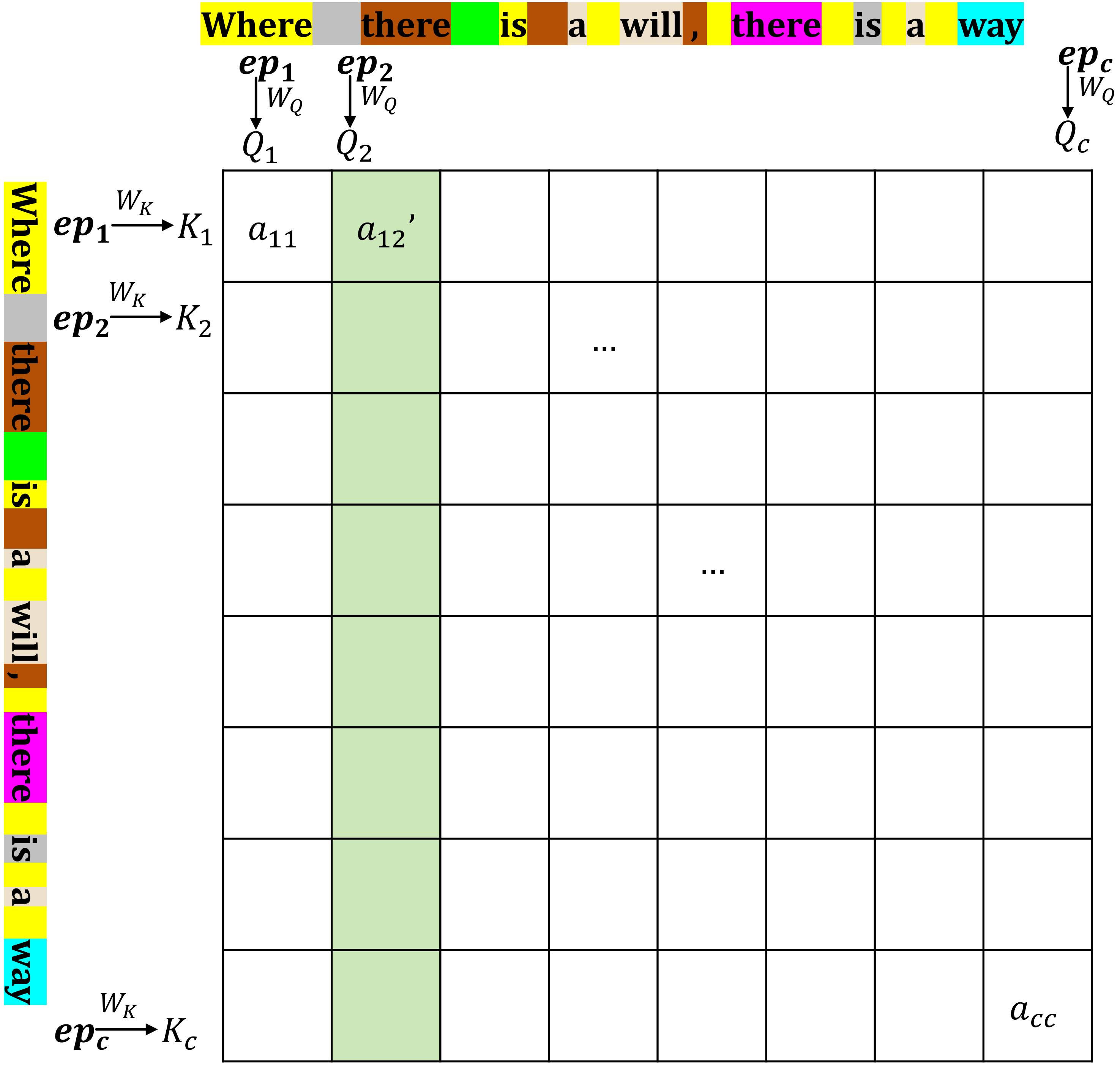


This is equivalent to masking the white region

For that, we make the entries at the white boxes $-\infty$

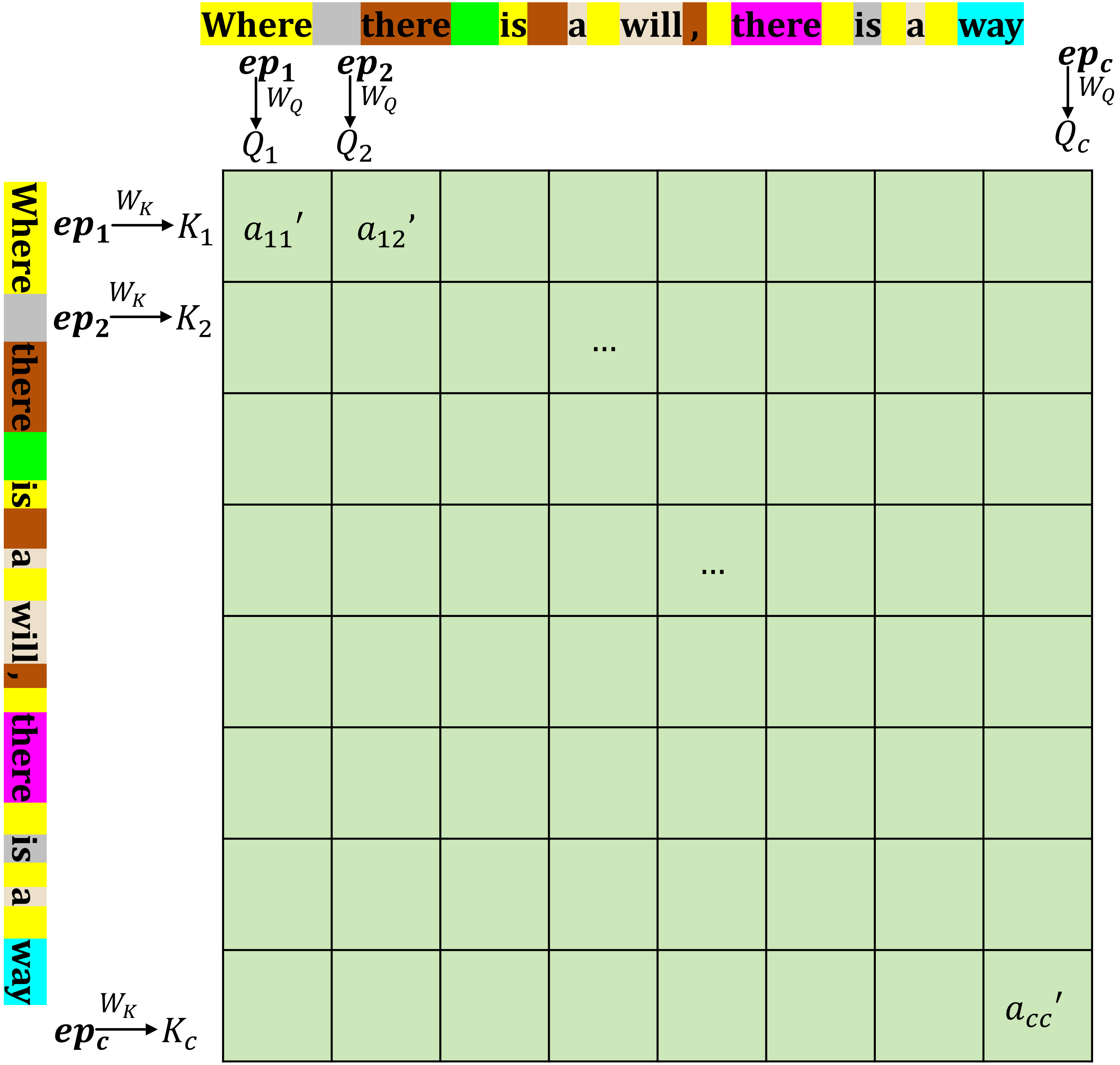
After column-wise softmax, these entries become 0

This is called **masked attention**



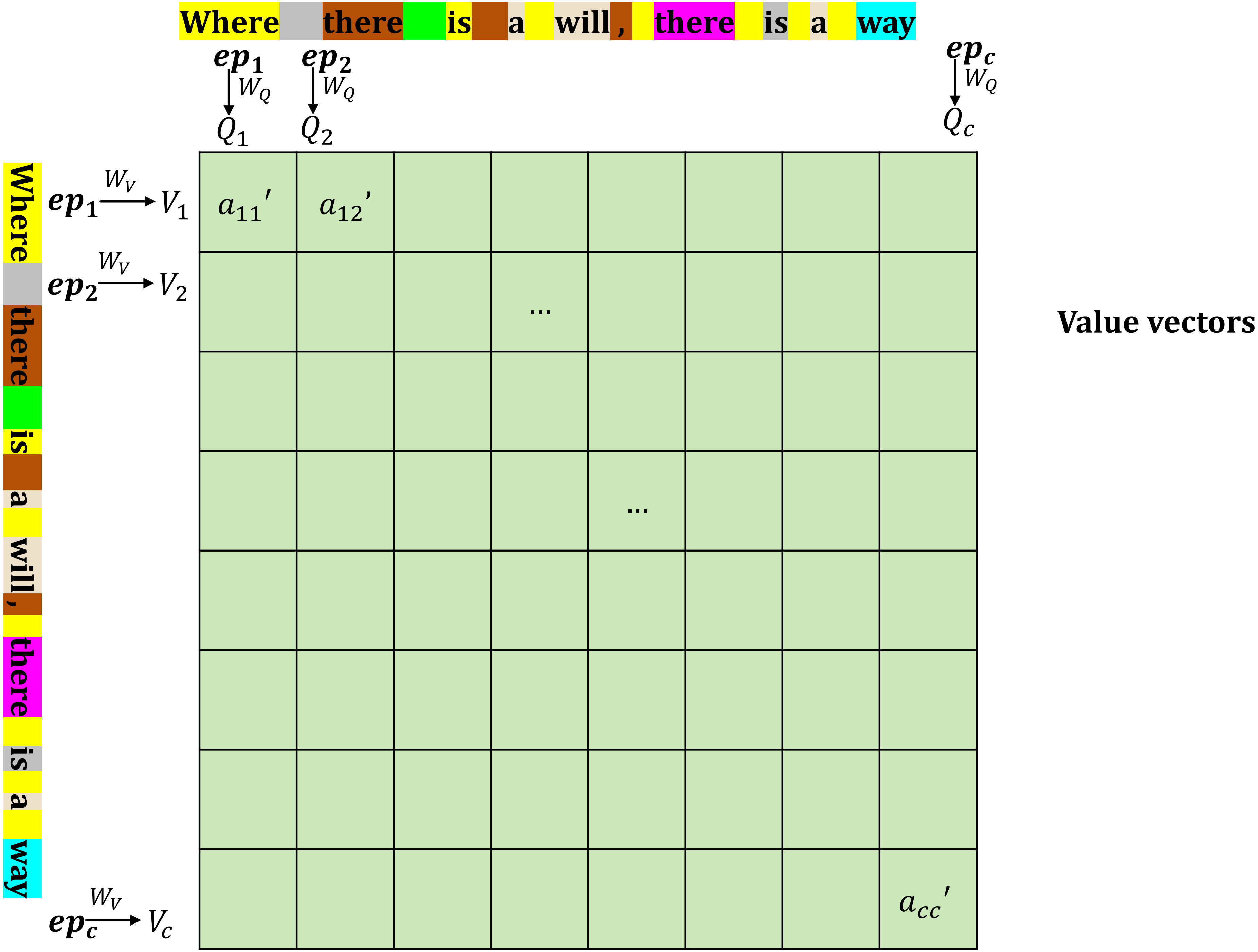
Attention matrix

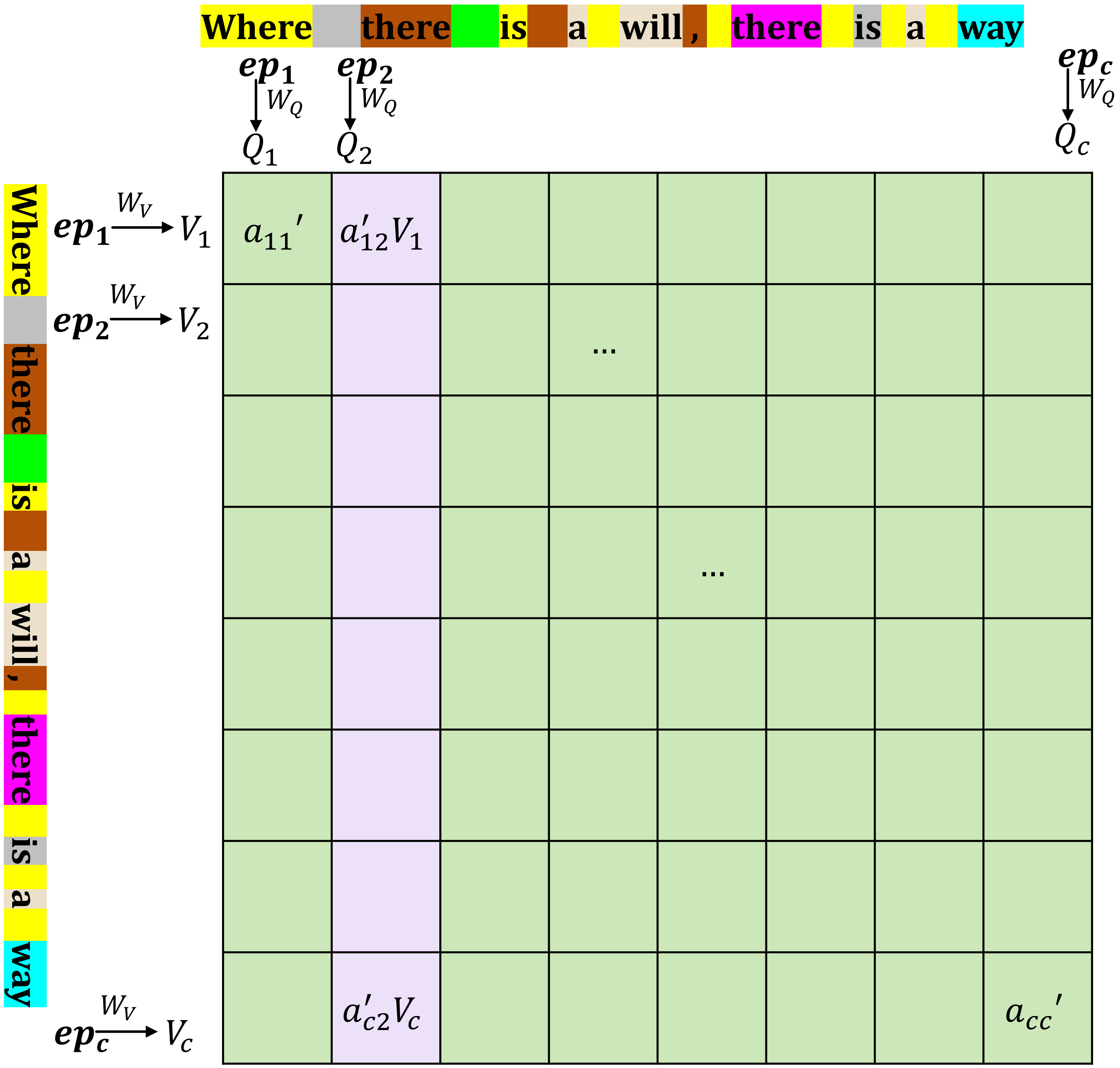
For each query column, we apply softmax. As a result, each column kind of indicates the probabilities of the corresponding query to match with different keys



Attention matrix

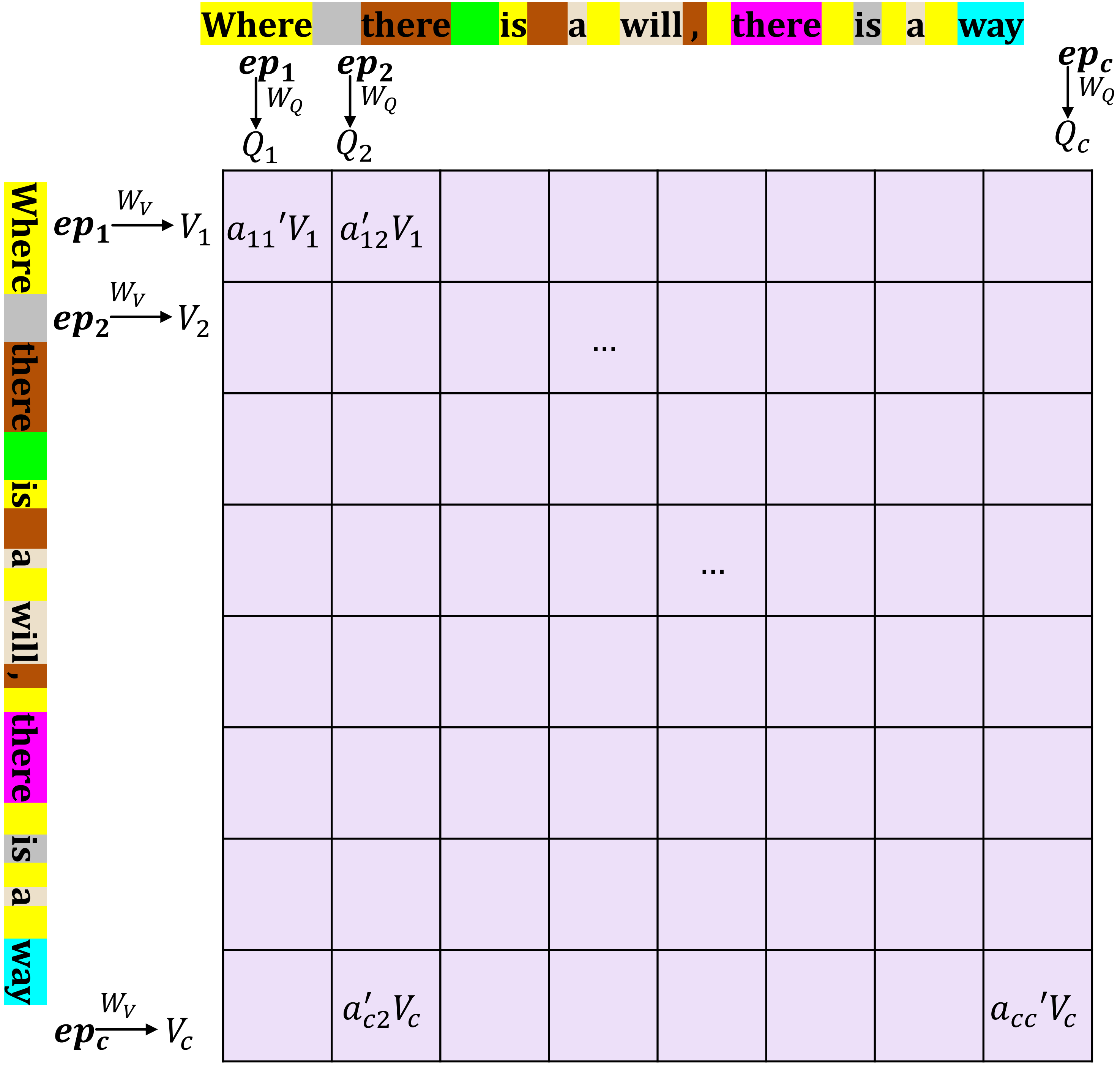
After applying softmax to every column


Where there is a will, there is a way
$ep_1 \xrightarrow{W_Q} Q_1$
$ep_2 \xrightarrow{W_Q} Q_2$
$ep_c \xrightarrow{W_Q} Q_c$
$ep_1 \xrightarrow{W_V} V_1$
$ep_2 \xrightarrow{W_V} V_2$
$ep_c \xrightarrow{W_V} V_c$
a_{11}'
a_{12}'
...
...
a_{cc}'
Value vectors



Multiply Value vectors with every column

Consider column 2



Multiply Value vectors with every column

Consider column 2

Similarly, for other columns

	$\boldsymbol{ep_1}$	$\boldsymbol{ep_2}$						$\boldsymbol{ep_c}$
$\boldsymbol{ep_1}$	$a_{11}{}'V_1$	$a'_{12}V_1$						
$\boldsymbol{ep_2}$				…				
					…			
$\boldsymbol{ep_c}$		$a'_{c2}V_c$						$a_{cc}{}'V_c$

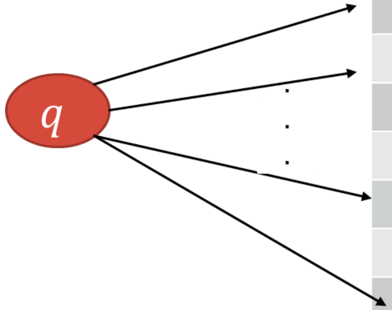
Multiply Value vectors with every column

Consider column 2

Similarly, for other columns

	ep_1	ep_2						ep_c
ep_1	$a_{11}'V_1$	$a_{12}'V_1$						
ep_2				…				
					…			
ep_c		$a_{c2}'V_c$						$a_{cc}'V_c$
		Δep_2						

Sum up values at each column



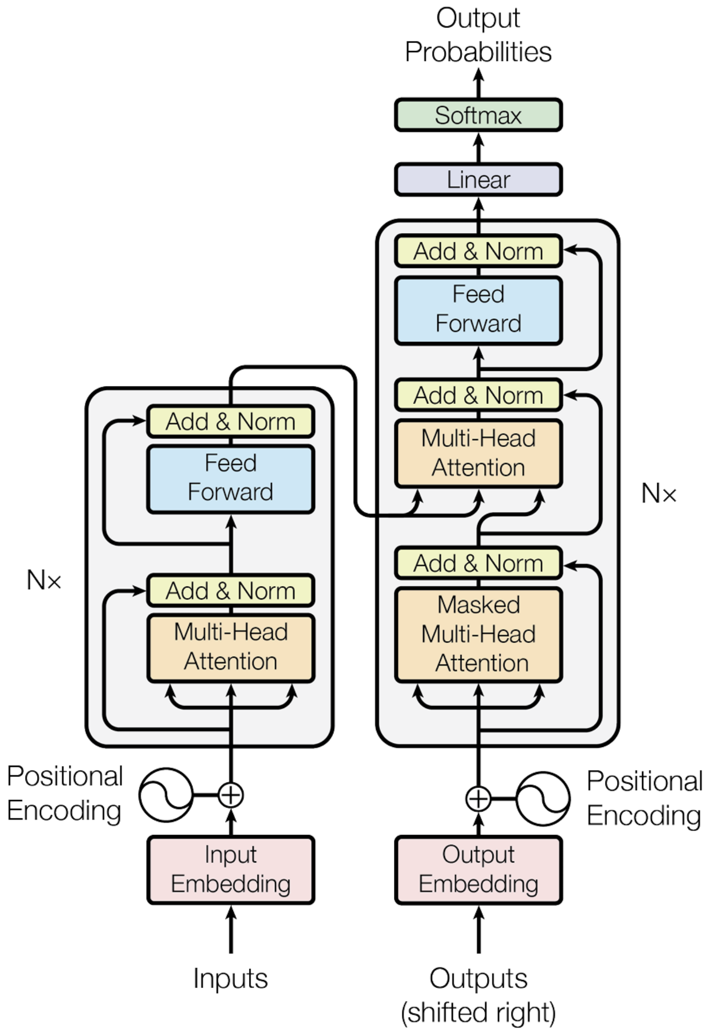
Similarity	Key	Value
s_1	k_1	v_1
s_2	k_2	v_2
.	.	.
.	.	.
s_m	k_m	v_m
.	.	.
s_n	k_n	v_n

$$Attention\ value = \sum_i softmax(s_i)v_i$$

	ep_1	ep_2						ep_c
ep_1	$a_{11}'V_1$	$a'_{12}V_1$						
ep_2				…				
					…			
ep_c		$a'_{c2}V_c$						$a_{cc}'V_c$
		$\boldsymbol{\Delta ep_2}$						

Sum up values at each column

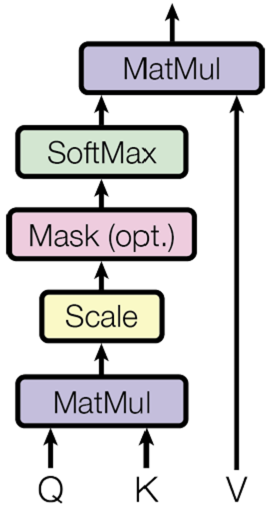
Get $ep'_i = ep_i + \Delta ep_i$
Through residual connection

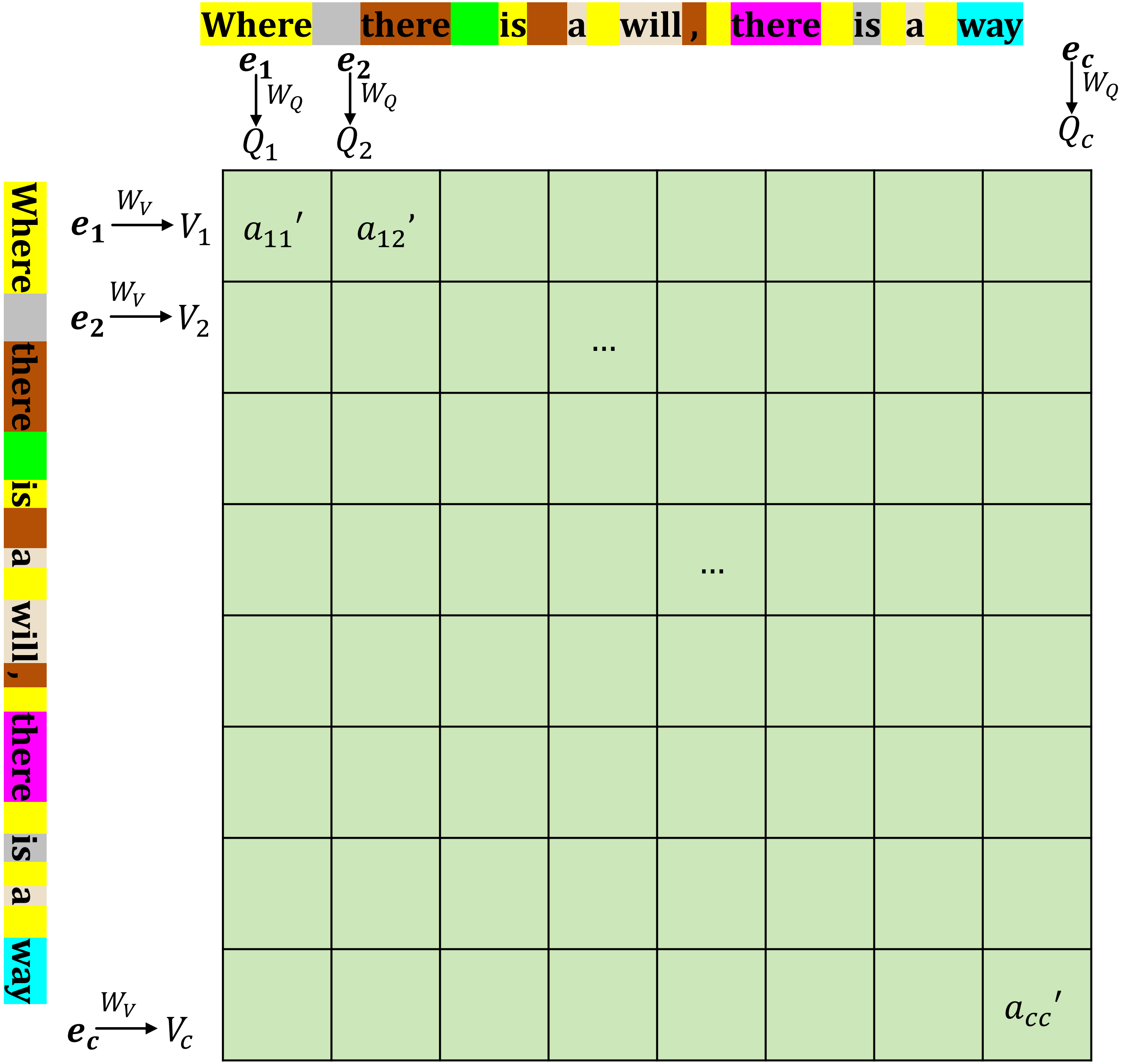


	ep_1	ep_2						ep_c
ep_1	$a_{11}'V_1$	$a_{12}'V_1$						
ep_2				...				
					...			
ep_c		$a_{c2}'V_c$						$a_{cc}'V_c$
		Δep_2						

This entire thing is one head of attention

Scaled Dot-Product Attention

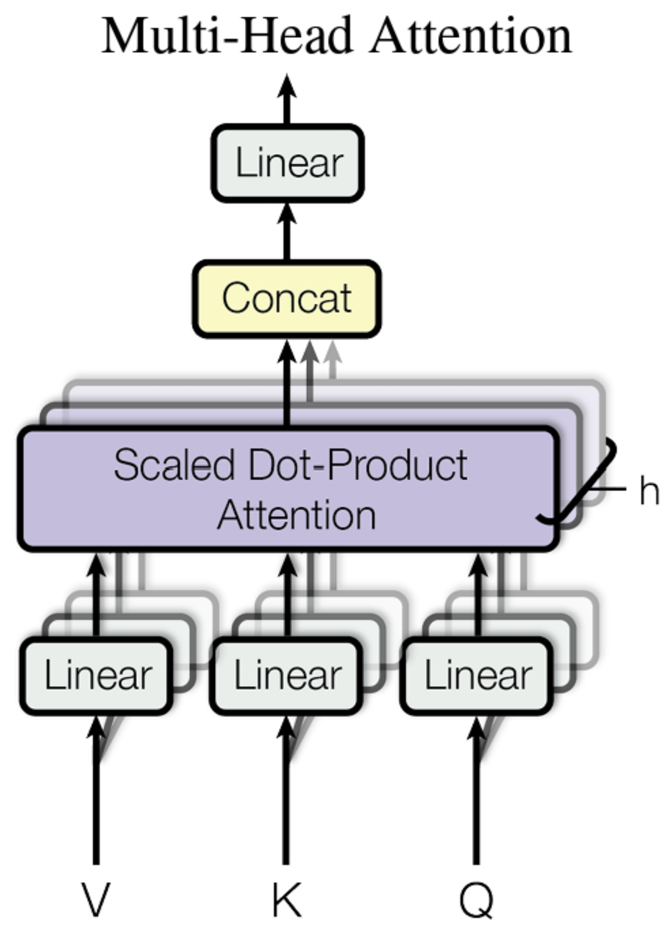


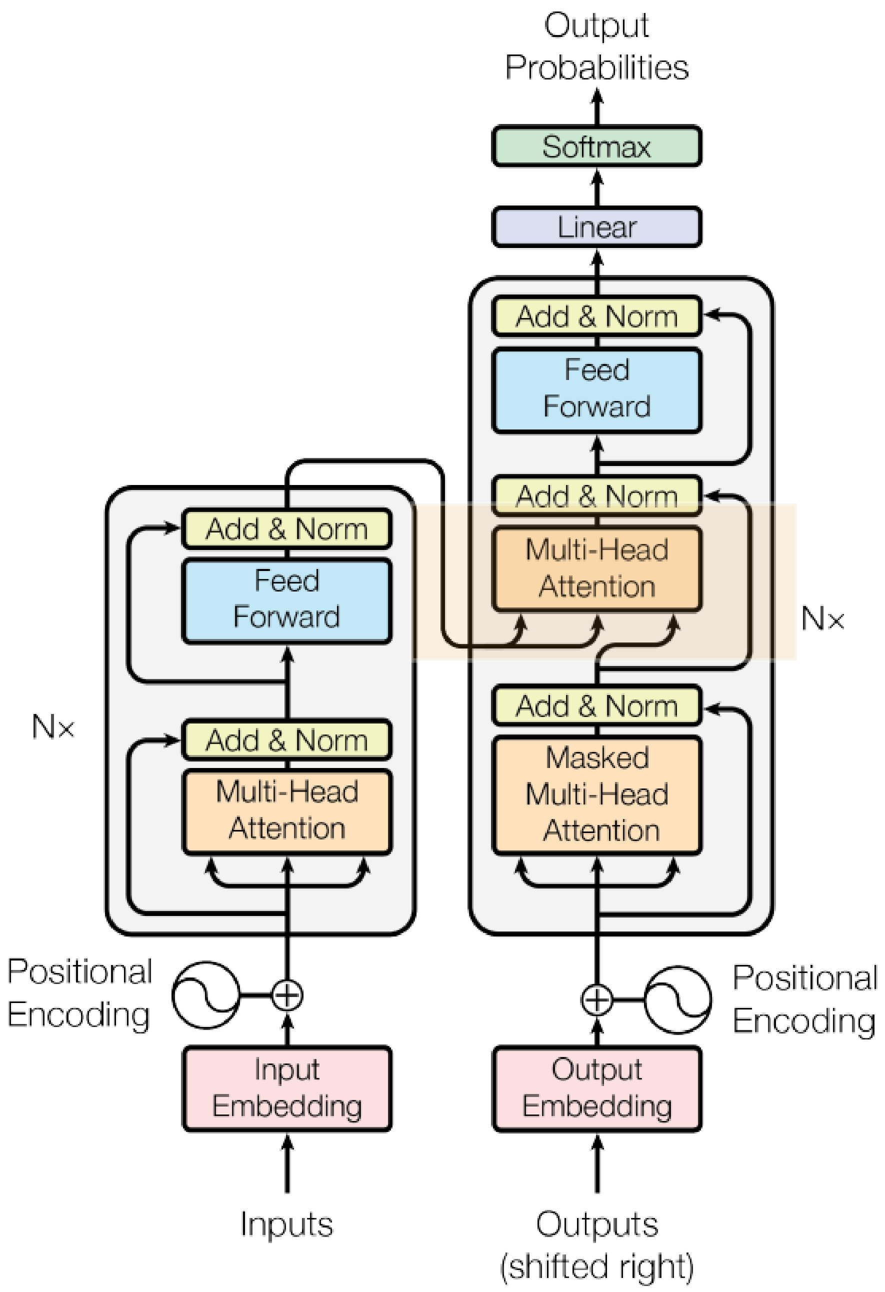


We will have many such heads with different $\boldsymbol{W_V}, \boldsymbol{W_Q}$, and $\boldsymbol{W_K}$

	$\boldsymbol{ep_1}$	$\boldsymbol{ep_2}$						$\boldsymbol{ep_c}$
$\boldsymbol{ep_1}$	$a_{11}{}'V_1$	$a'_{12}V_1$						
$\boldsymbol{ep_2}$				…				
					…			
$\boldsymbol{ep_c}$		$a'_{c2}V_c$						$a_{cc}{}'V_c$
		$\boldsymbol{\Delta ep_2}$						

This entire thing is one head of attention



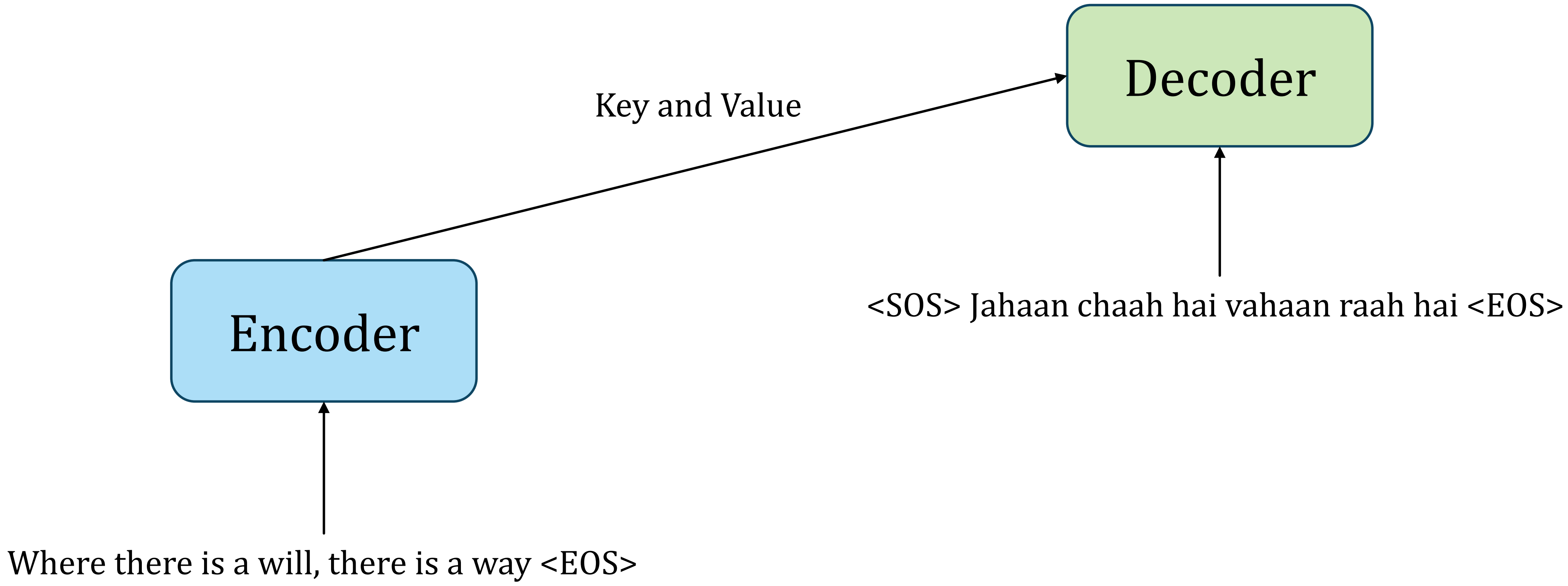


Encoder-decoder Attention

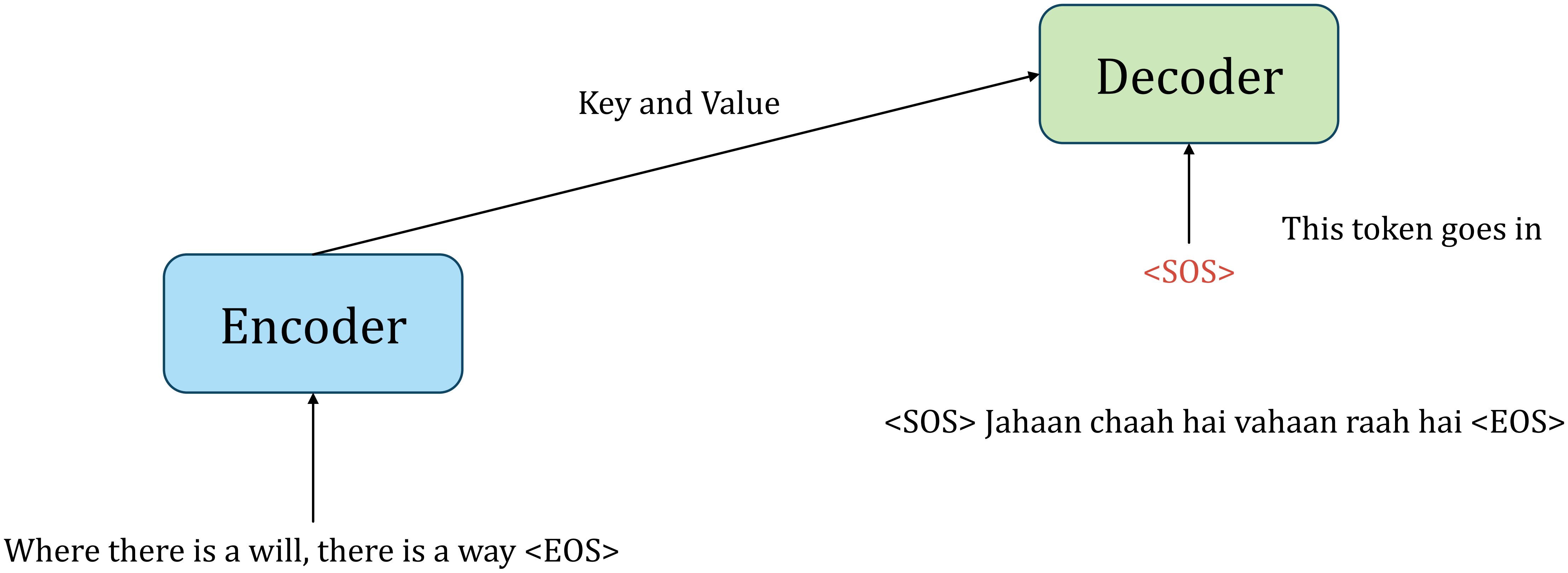
Key and value comes from the encoder through two paths

Query comes from the decoder

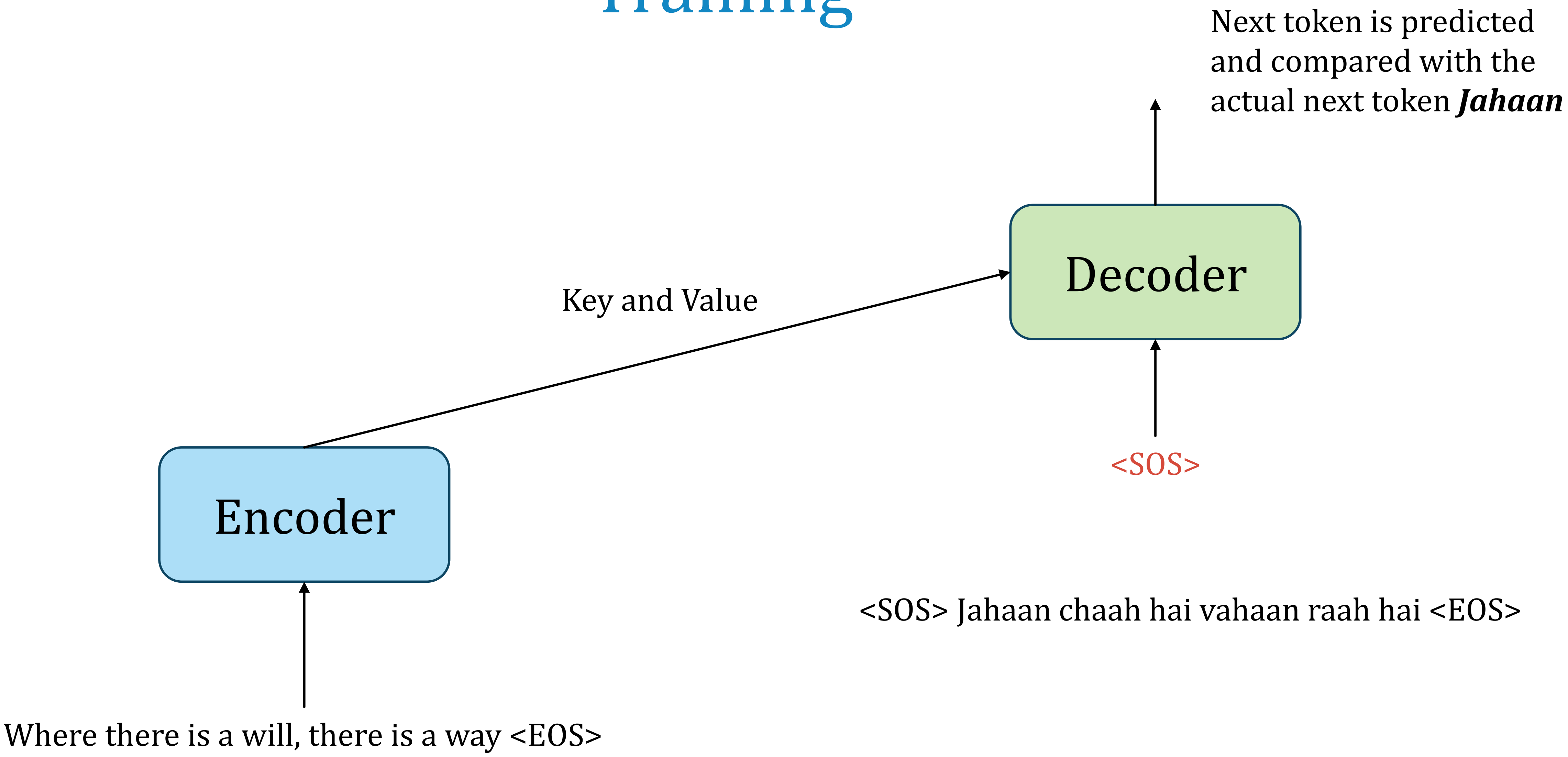
Training


Decoder
Key and Value
Encoder
<SOS> Jahaan chaah hai vahaan raah hai <EOS>
Where there is a will, there is a way <EOS>

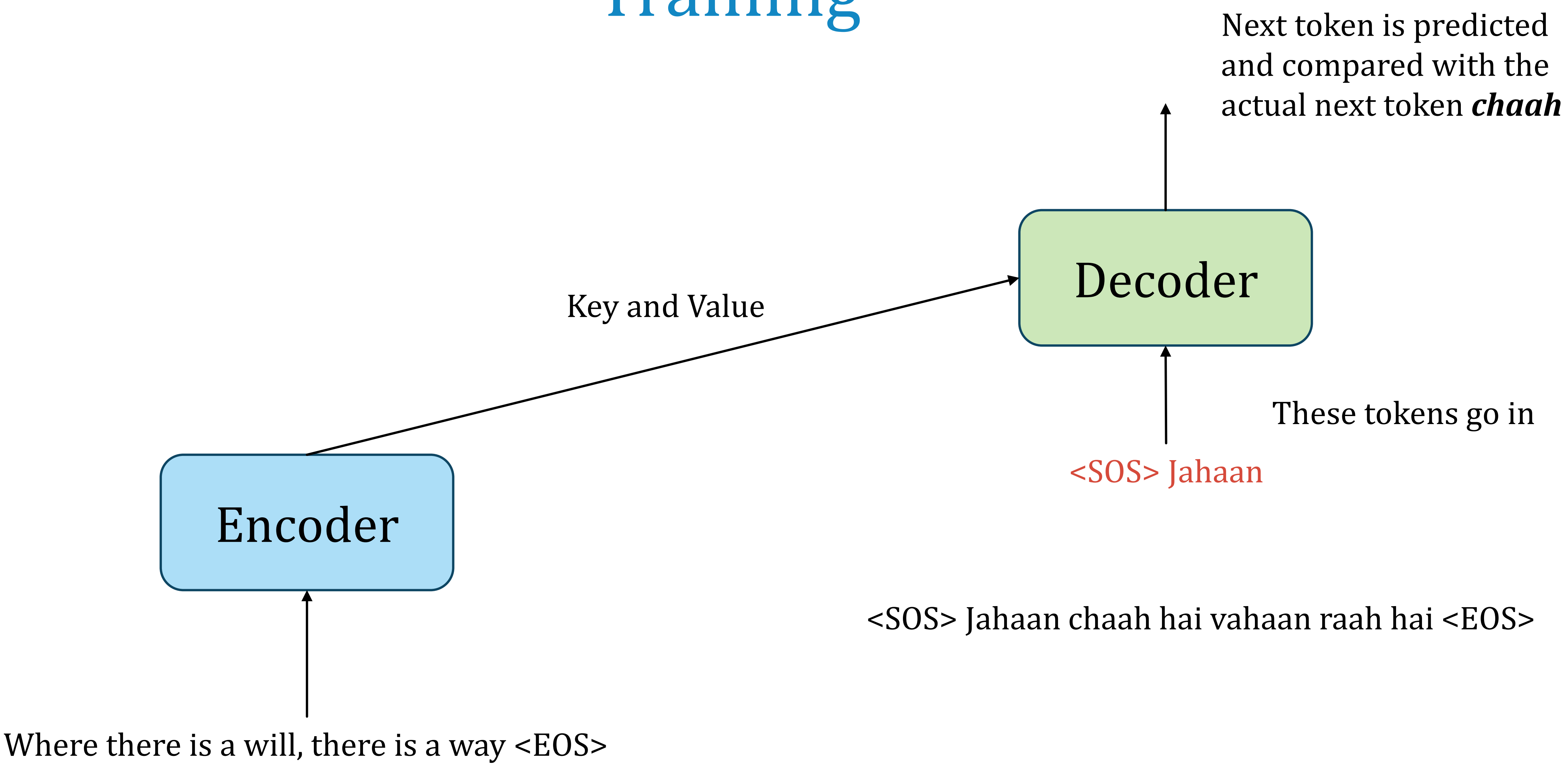
Training


Decoder
Key and Value
This token goes in
<SOS>
Encoder
<SOS> Jahaan chaah hai vahaan raah hai <EOS>
Where there is a will, there is a way <EOS>

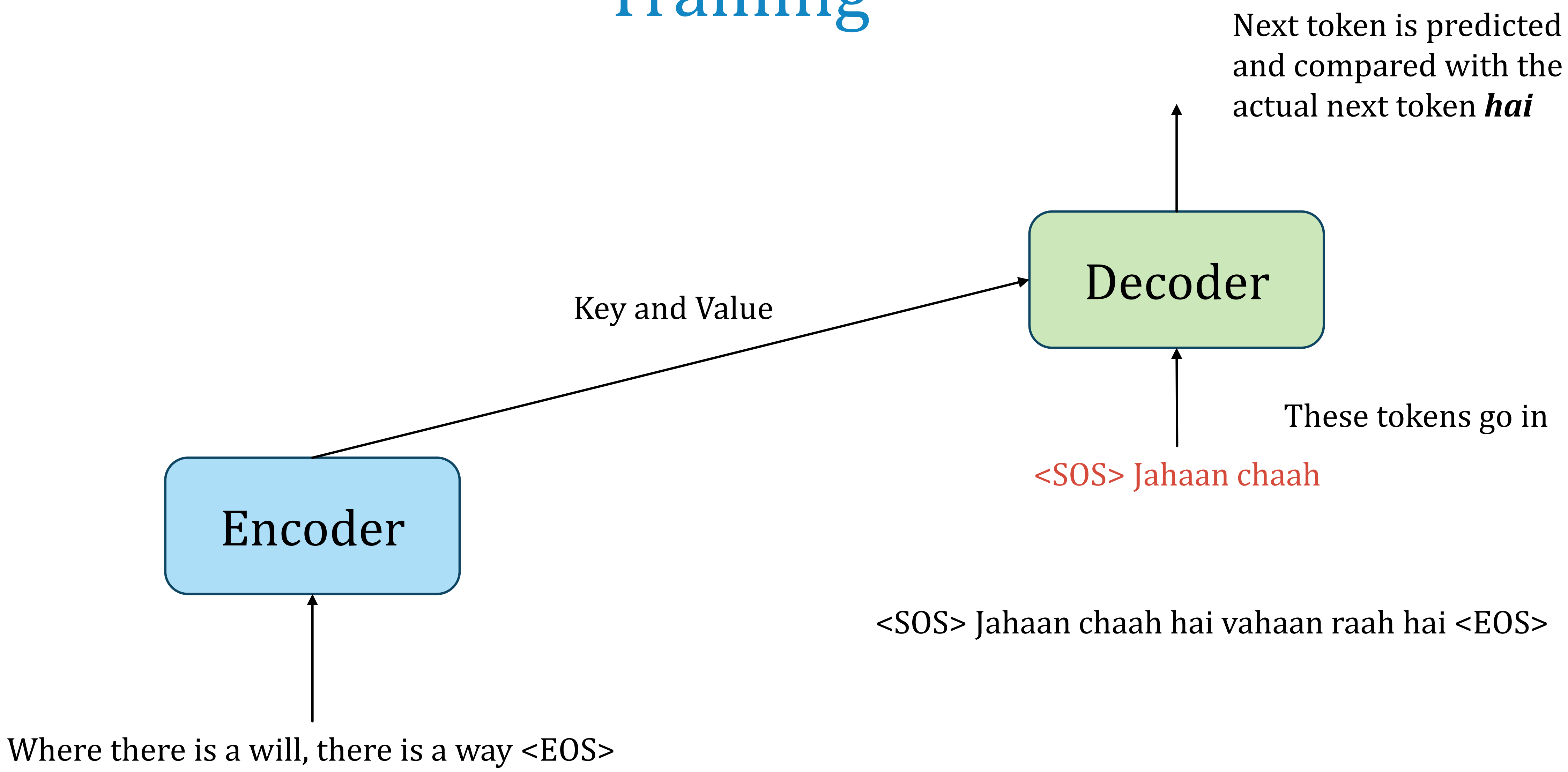
Training


Next token is predicted and compared with the actual next token ***Jahaan***
Decoder
Key and Value
<SOS>
Encoder
<SOS> Jahaan chaah hai vahaan raah hai <EOS>
Where there is a will, there is a way <EOS>

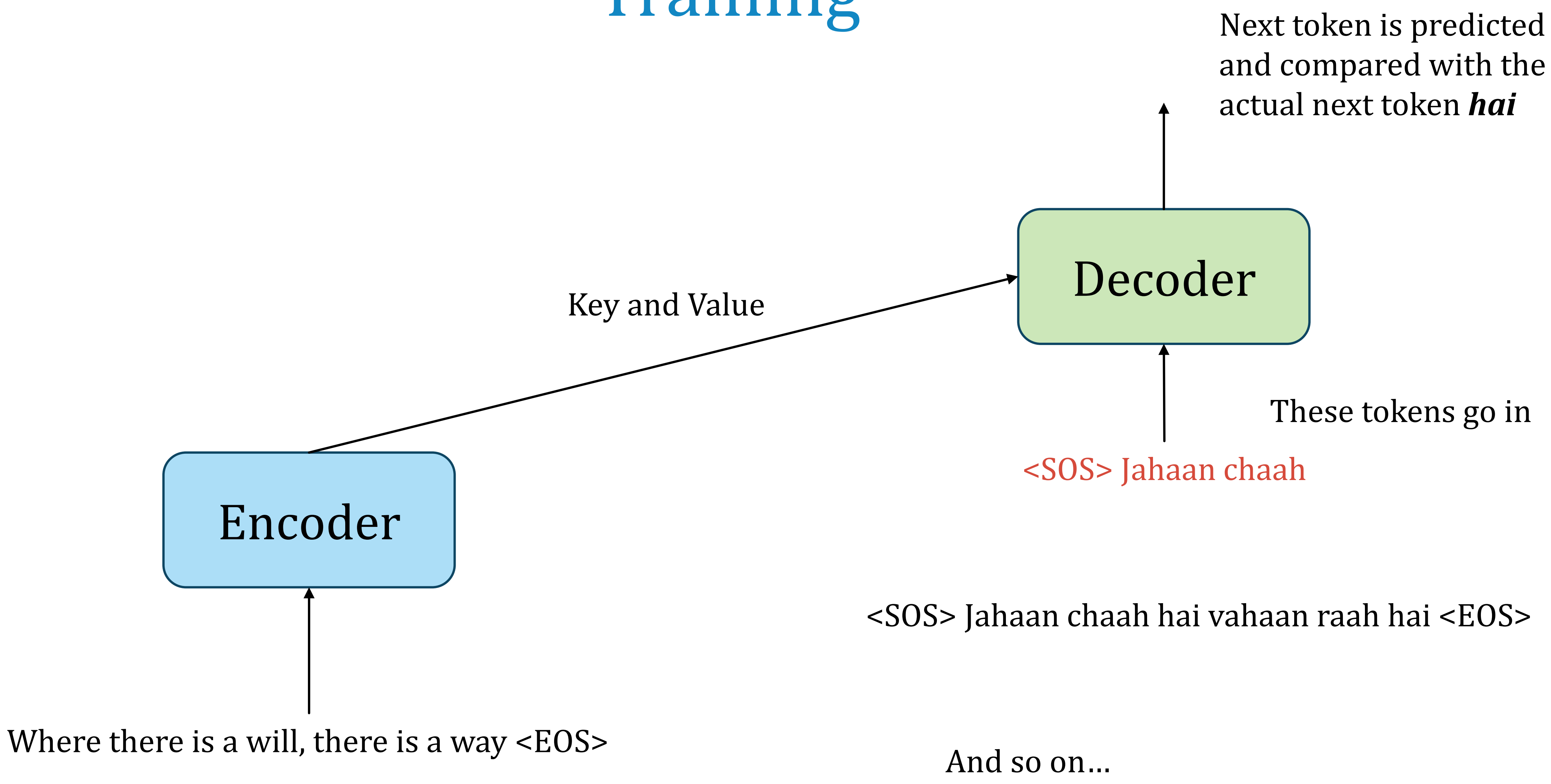
Training


Next token is predicted and compared with the actual next token ***chaah***
Decoder
Key and Value
These tokens go in
<SOS> Jahaan
Encoder
<SOS> Jahaan chaah hai vahaan raah hai <EOS>
Where there is a will, there is a way <EOS>

Training


Next token is predicted
and compared with the
actual next token ***hai***
Decoder
Key and Value
These tokens go in
<SOS> Jahaan chaah
Encoder
<SOS> Jahaan chaah hai vahaan raah hai <EOS>
Where there is a will, there is a way <EOS>

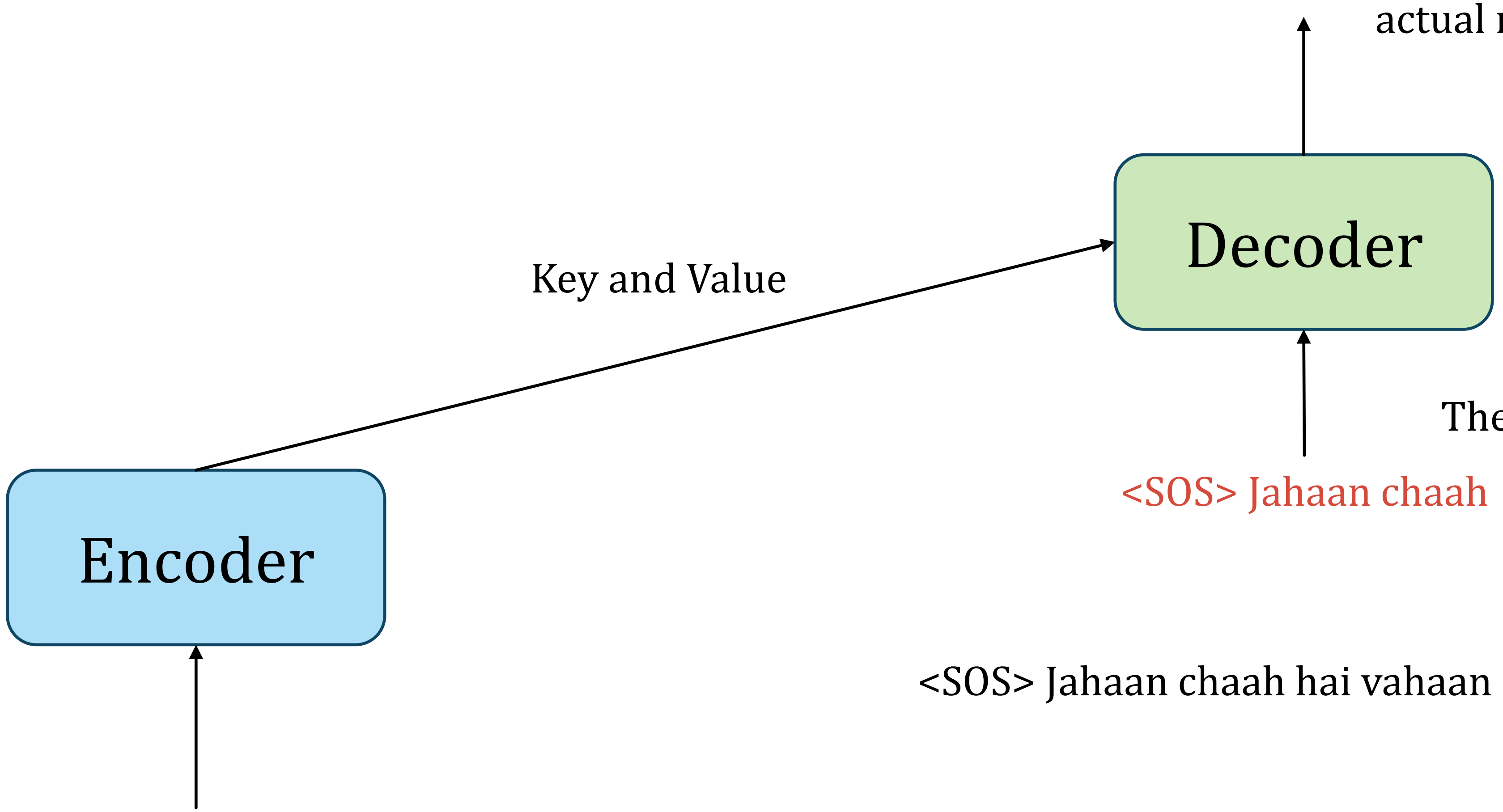
Training



Next token is predicted and compared with the actual next token ***hai***
Key and Value
Decoder
These tokens go in
<SOS> Jahaan chaah
Encoder
<SOS> Jahaan chaah hai vahaan raah hai <EOS>
Where there is a will, there is a way <EOS>
And so on…

Training

Next token is predicted and compared with the actual next token ***hai***



These tokens go in

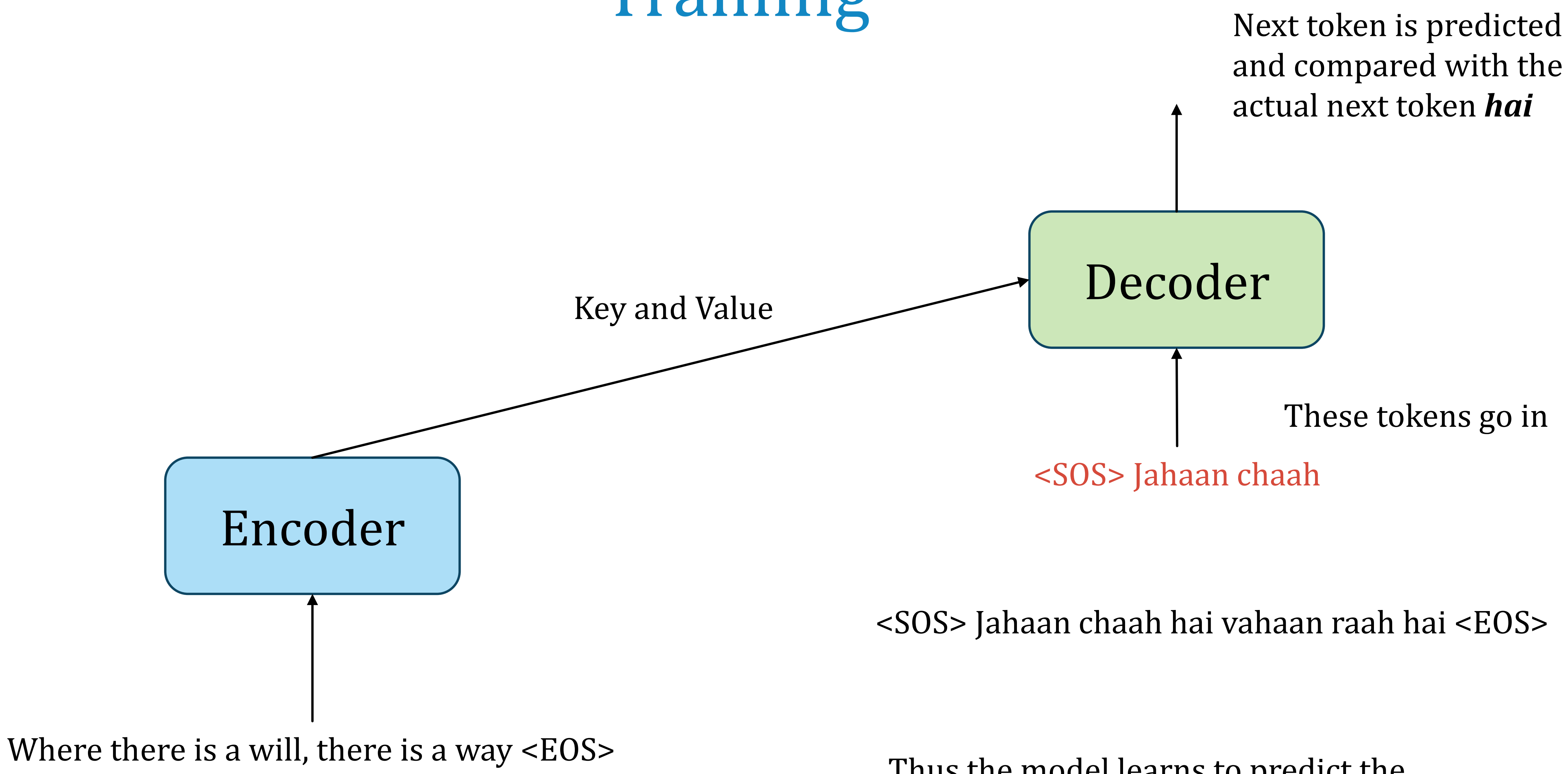
<SOS> Jahaan chaah

<SOS> Jahaan chaah hai vahaan raah hai <EOS>

Where there is a will, there is a way <EOS>

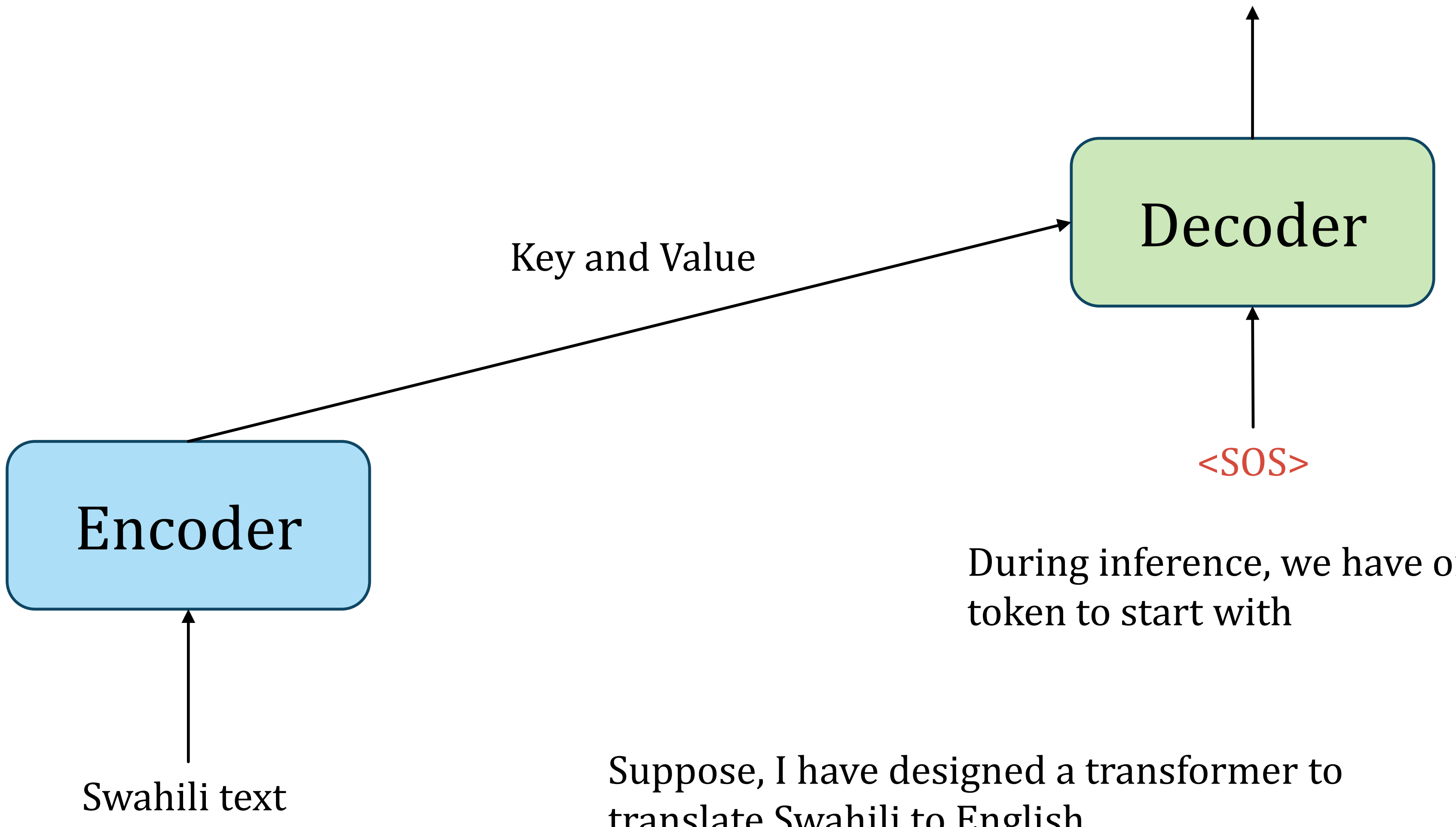
We may calculate CE loss and backpropagate

Training



Thus the model learns to predict the next token given the previous tokens

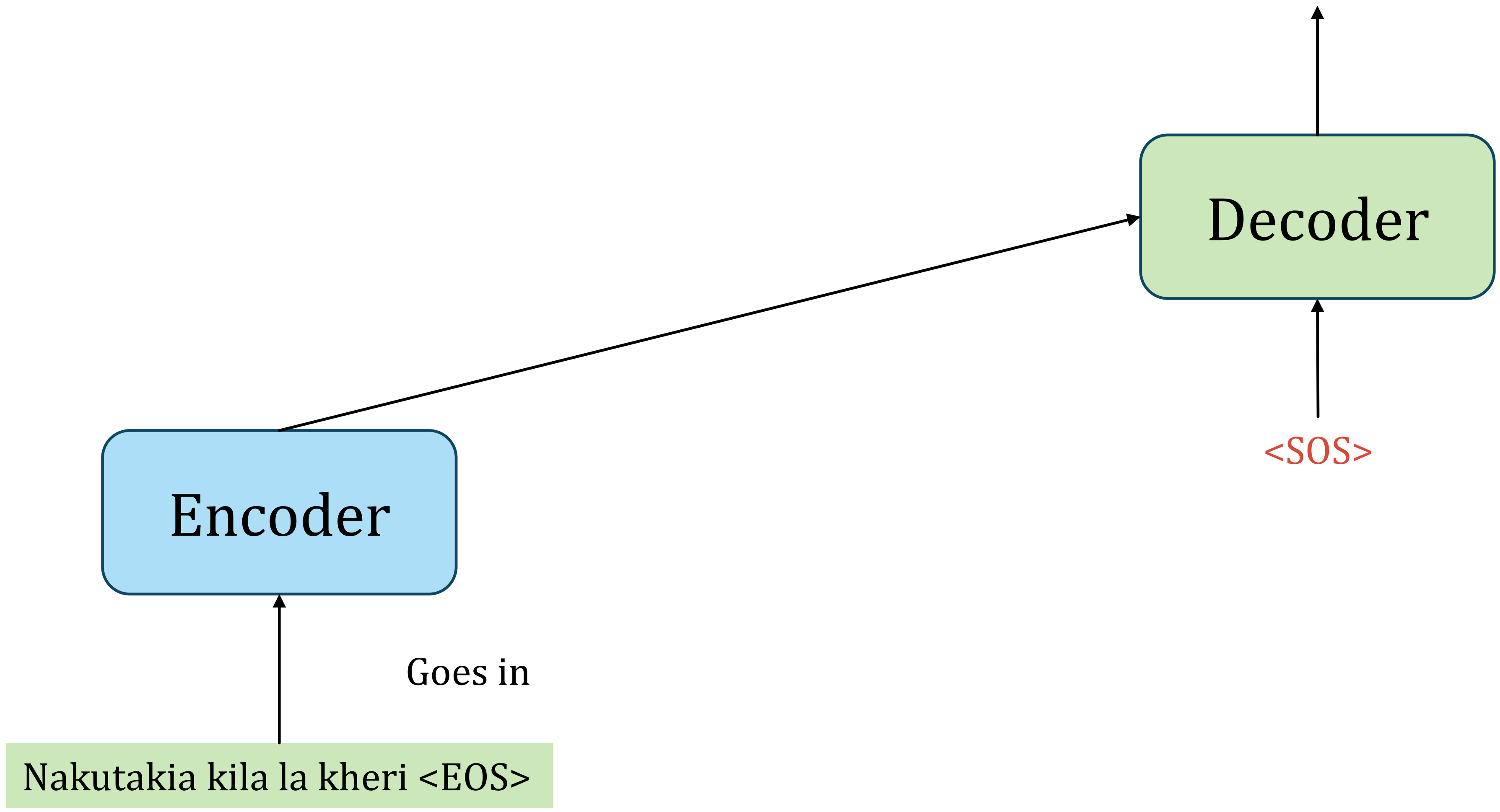
Inference



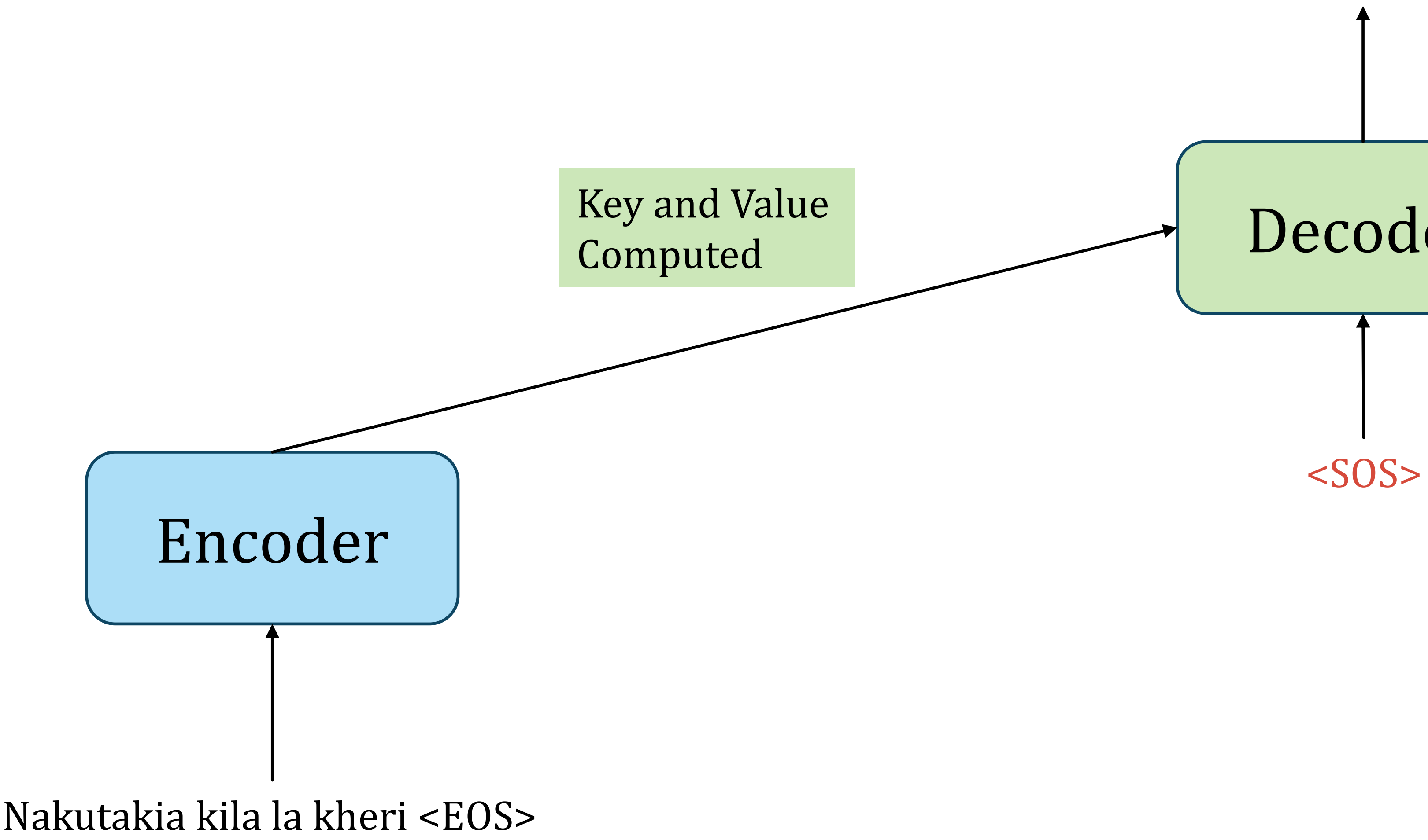
During inference, we have only SOS token to start with

Suppose, I have designed a transformer to translate Swahili to English

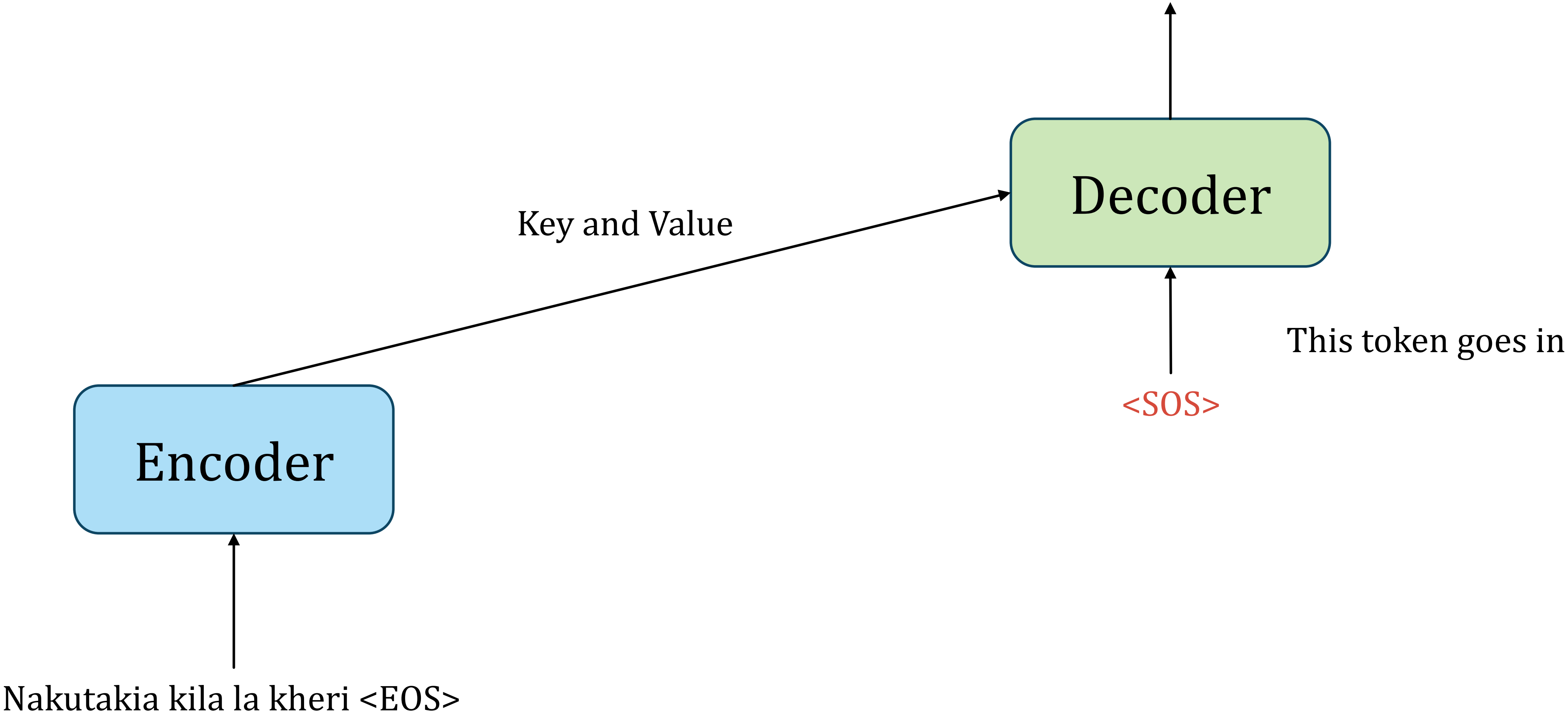
Inference



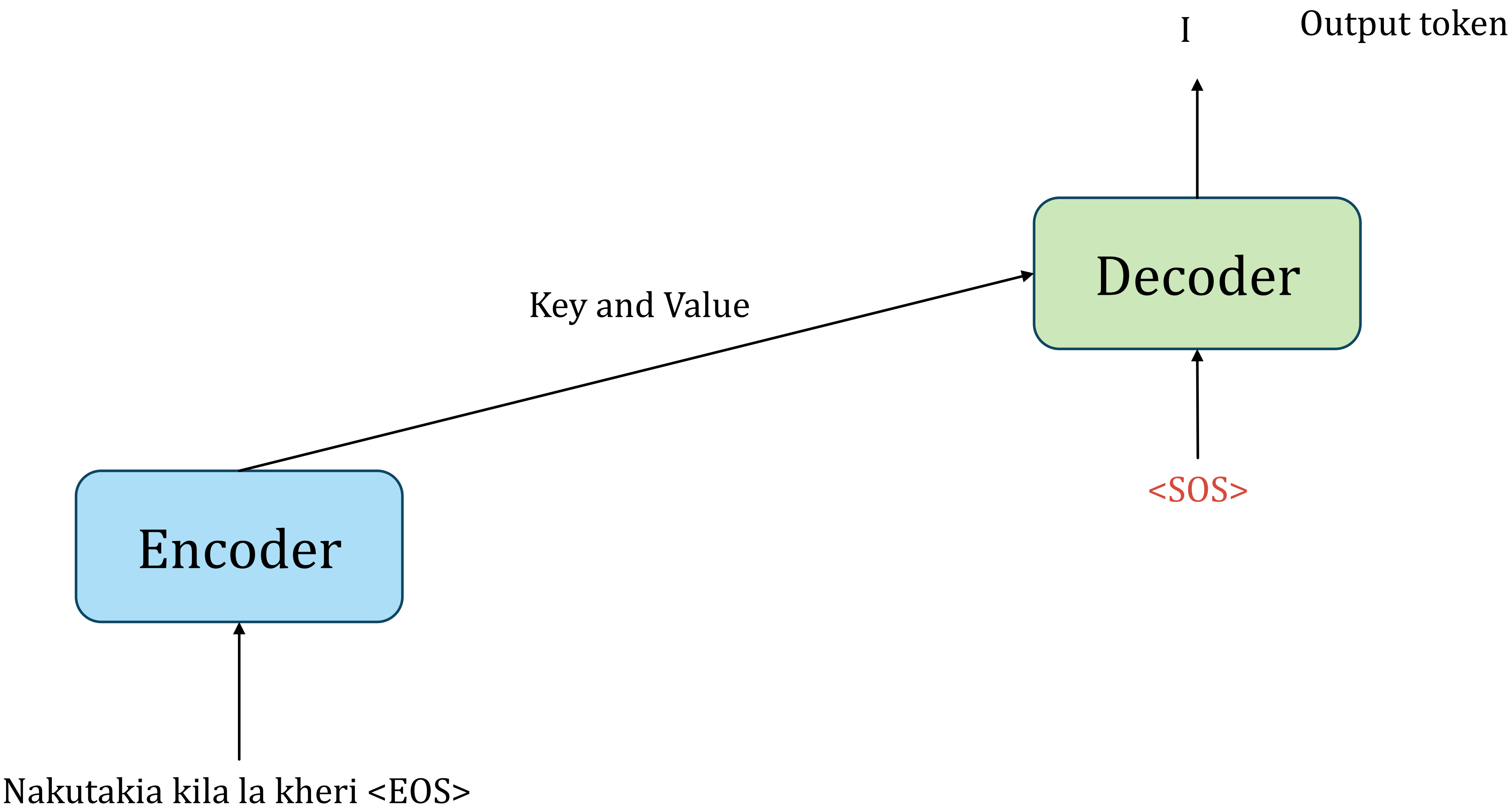
Inference


Key and Value
Computed
Decoder
<SOS>
Encoder
Nakutakia kila la kheri <EOS>

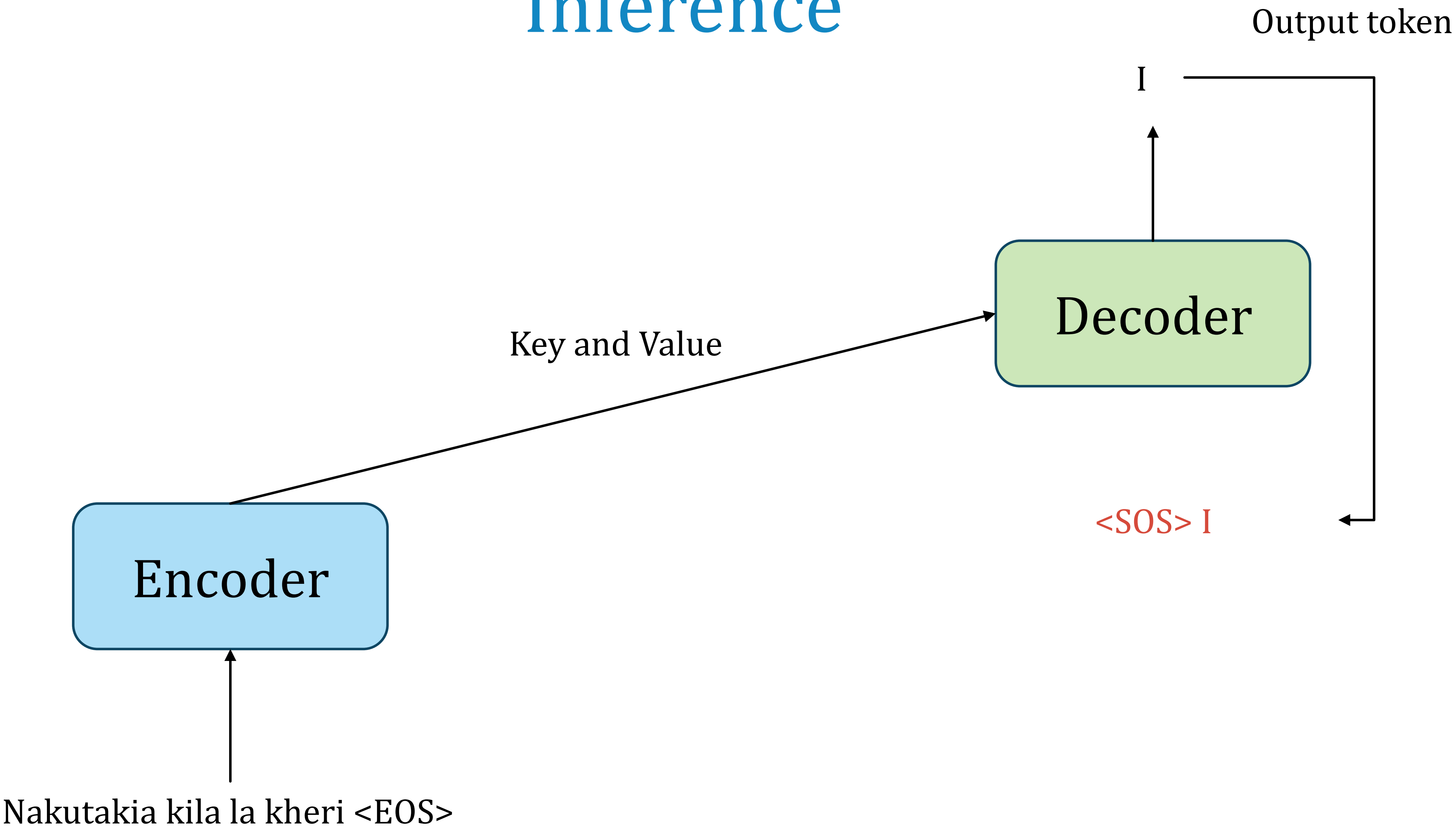
Inference


Decoder
Key and Value
This token goes in
<SOS>
Encoder
Nakutakia kila la kheri <EOS>

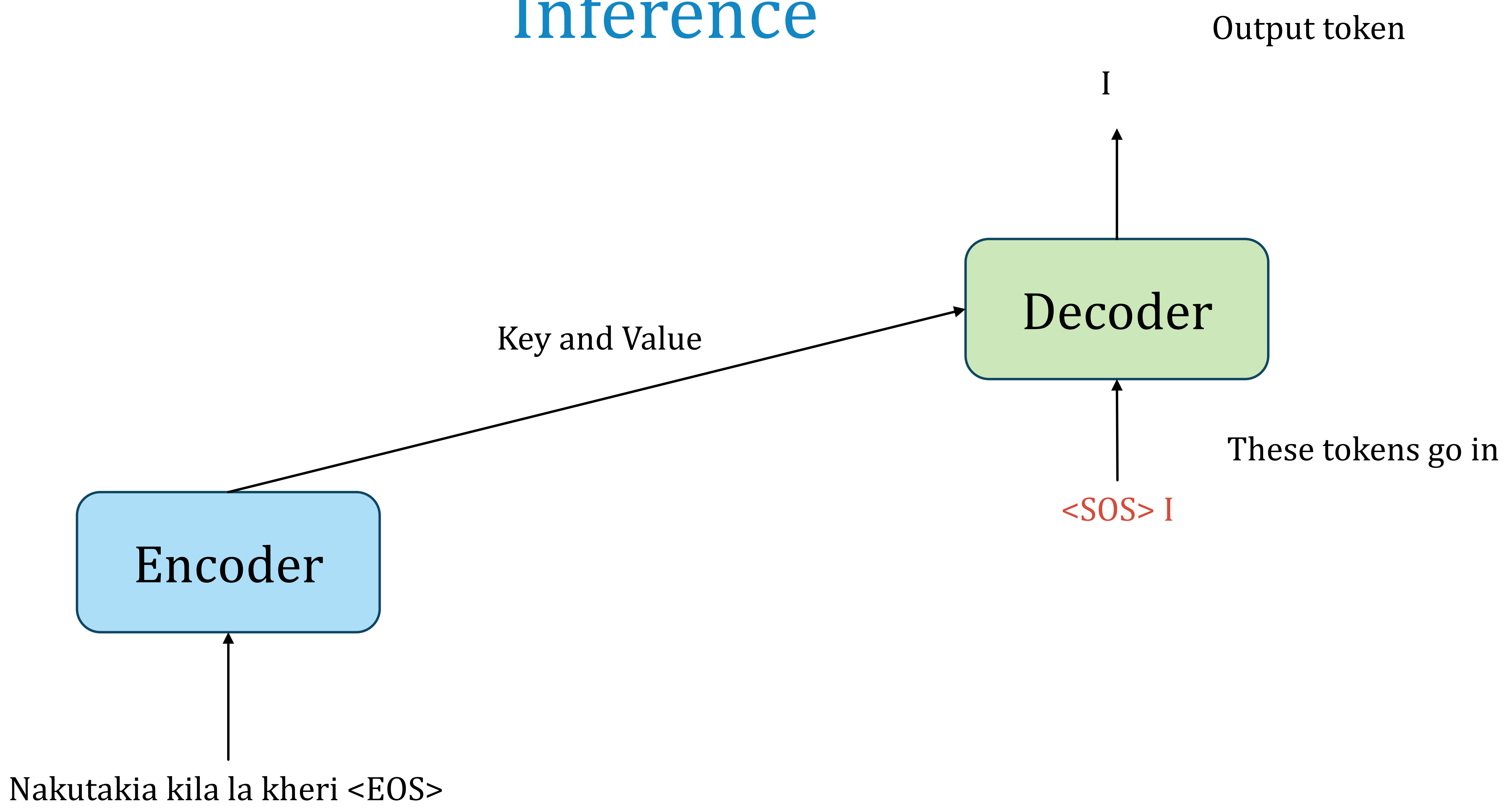
Inference


I
Output token
Decoder
Key and Value
<SOS>
Encoder
Nakutakia kila la kheri <EOS>

Inference

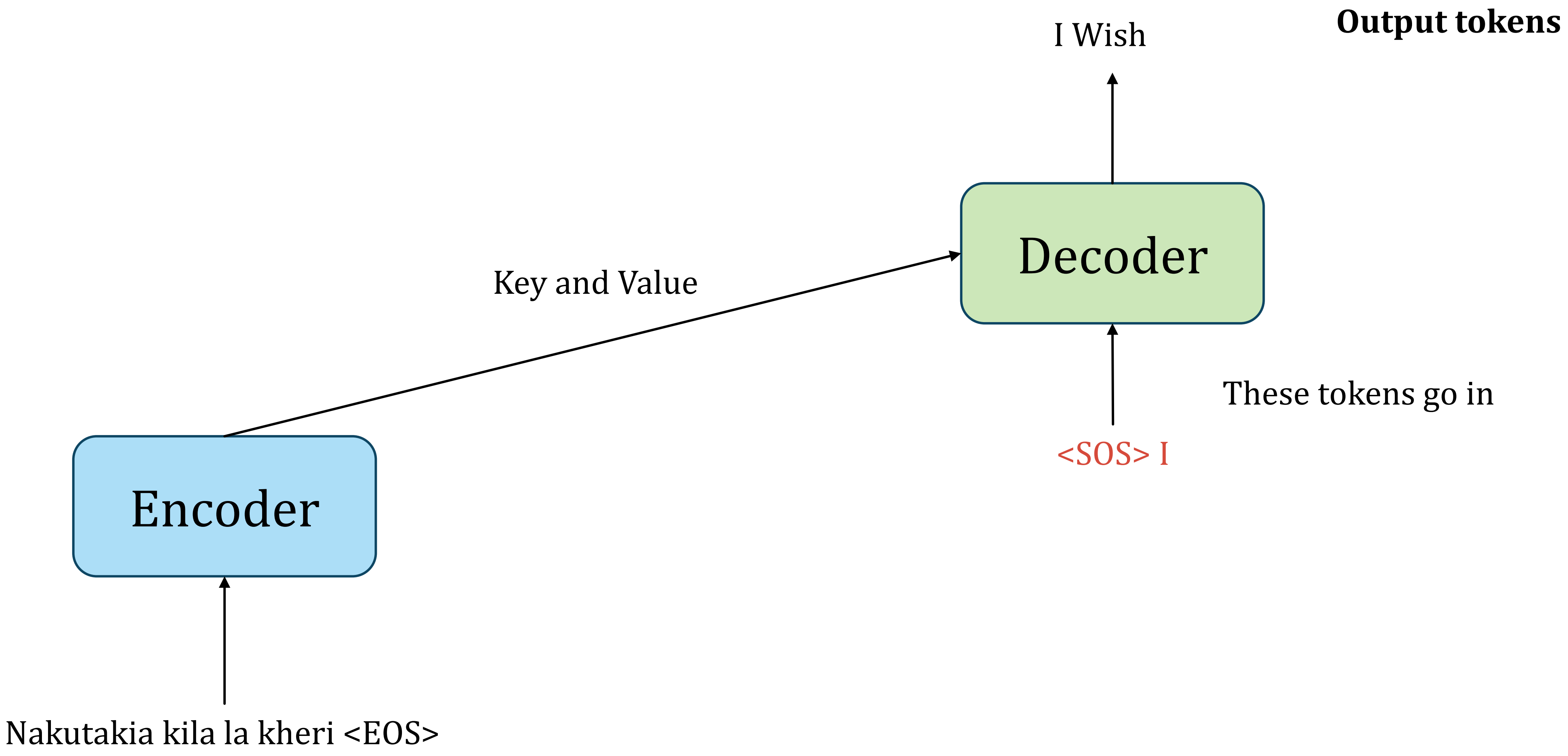

Output token
I
Decoder
Key and Value
<SOS> I
Encoder
Nakutakia kila la kheri <EOS>

Inference



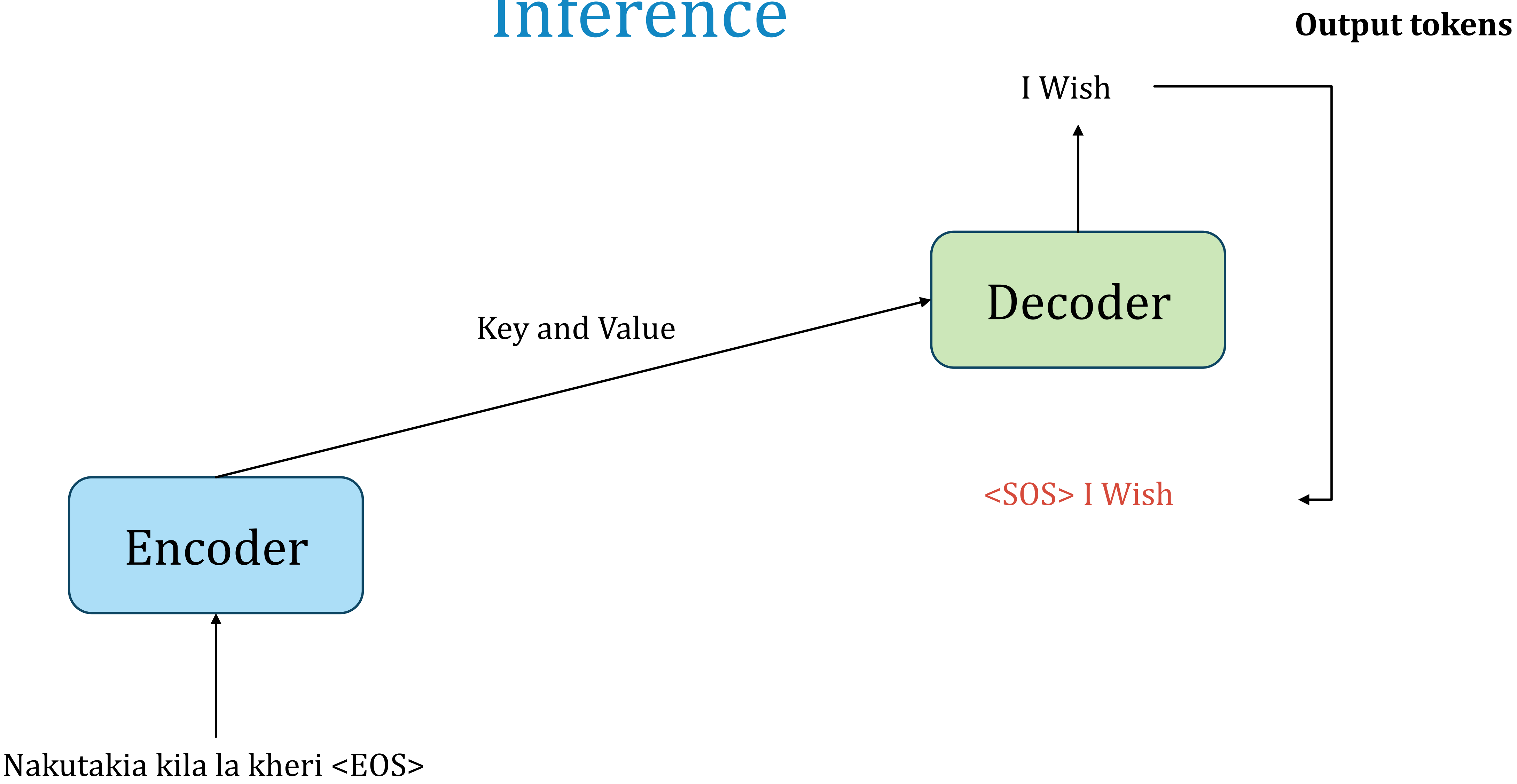
Output token
I
Decoder
Key and Value
These tokens go in
<SOS> I
Encoder
Nakutakia kila la kheri <EOS>

Inference

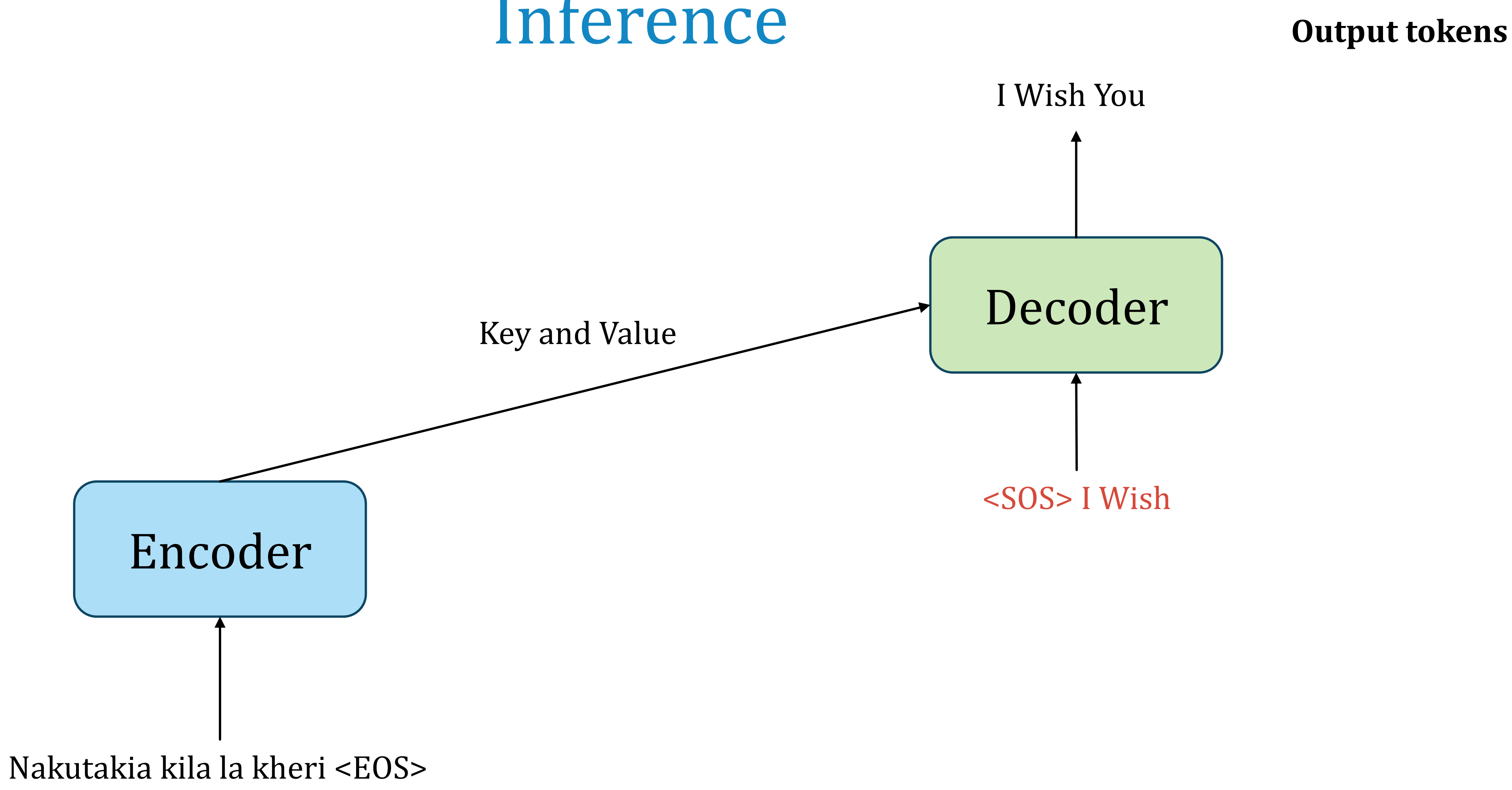


Output tokens
I Wish
Decoder
Key and Value
These tokens go in
<SOS> I
Encoder
Nakutakia kila la kheri <EOS>

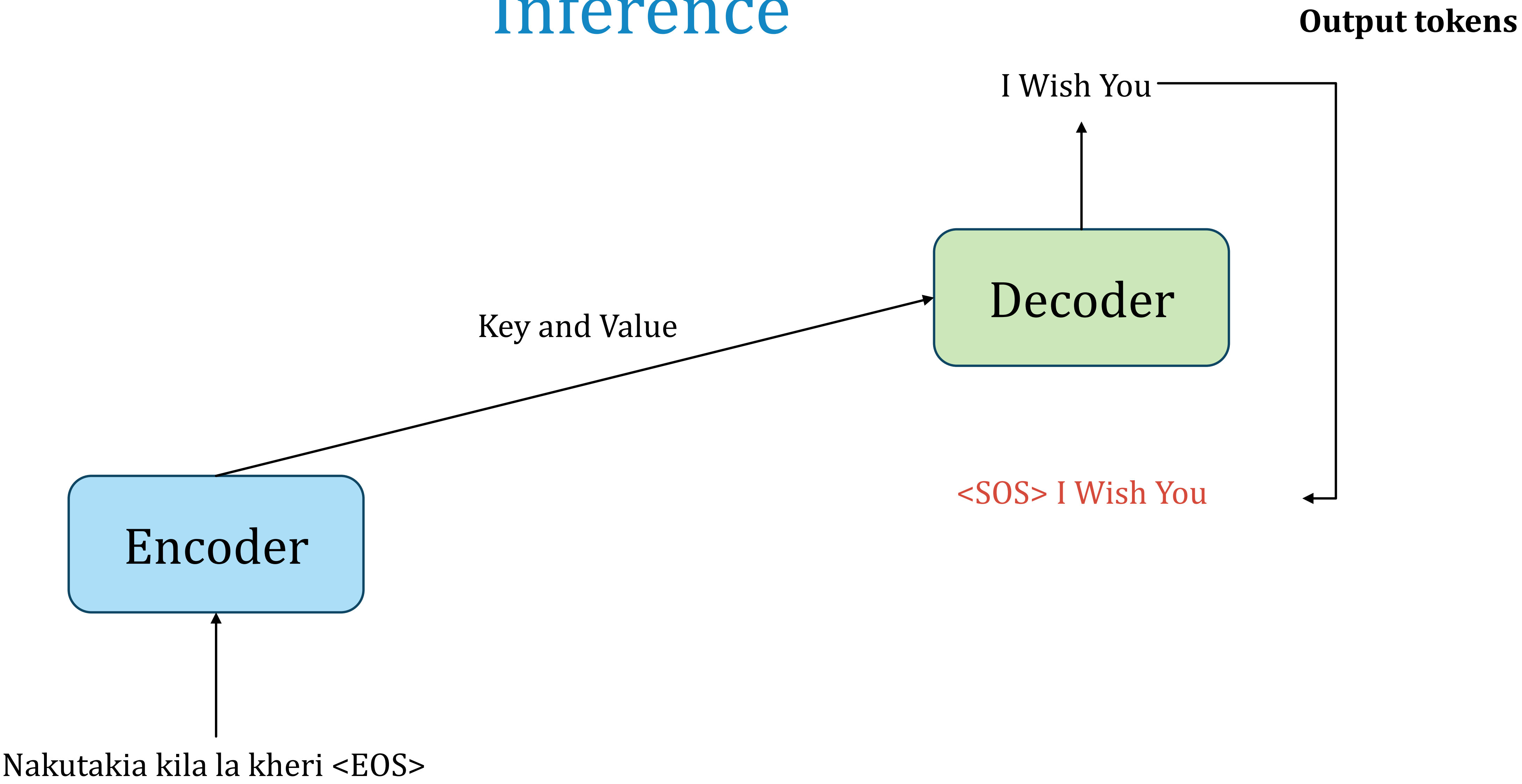
Inference


Output tokens
I Wish
Decoder
Key and Value
<SOS> I Wish
Encoder
Nakutakia kila la kheri <EOS>

Inference

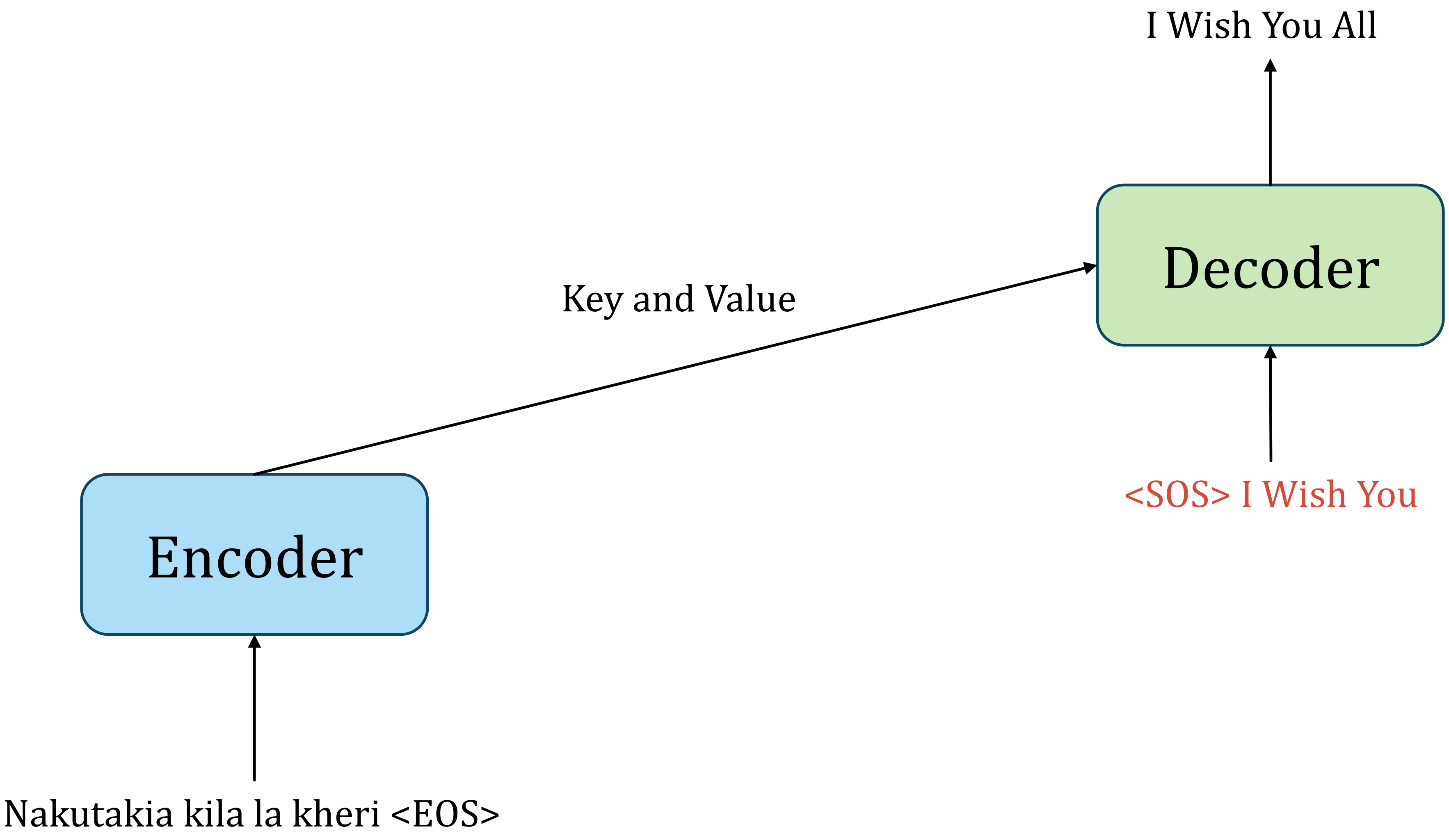

Output tokens
I Wish You
Decoder
Key and Value
Encoder
<SOS> I Wish
Nakutakia kila la kheri <EOS>

Inference

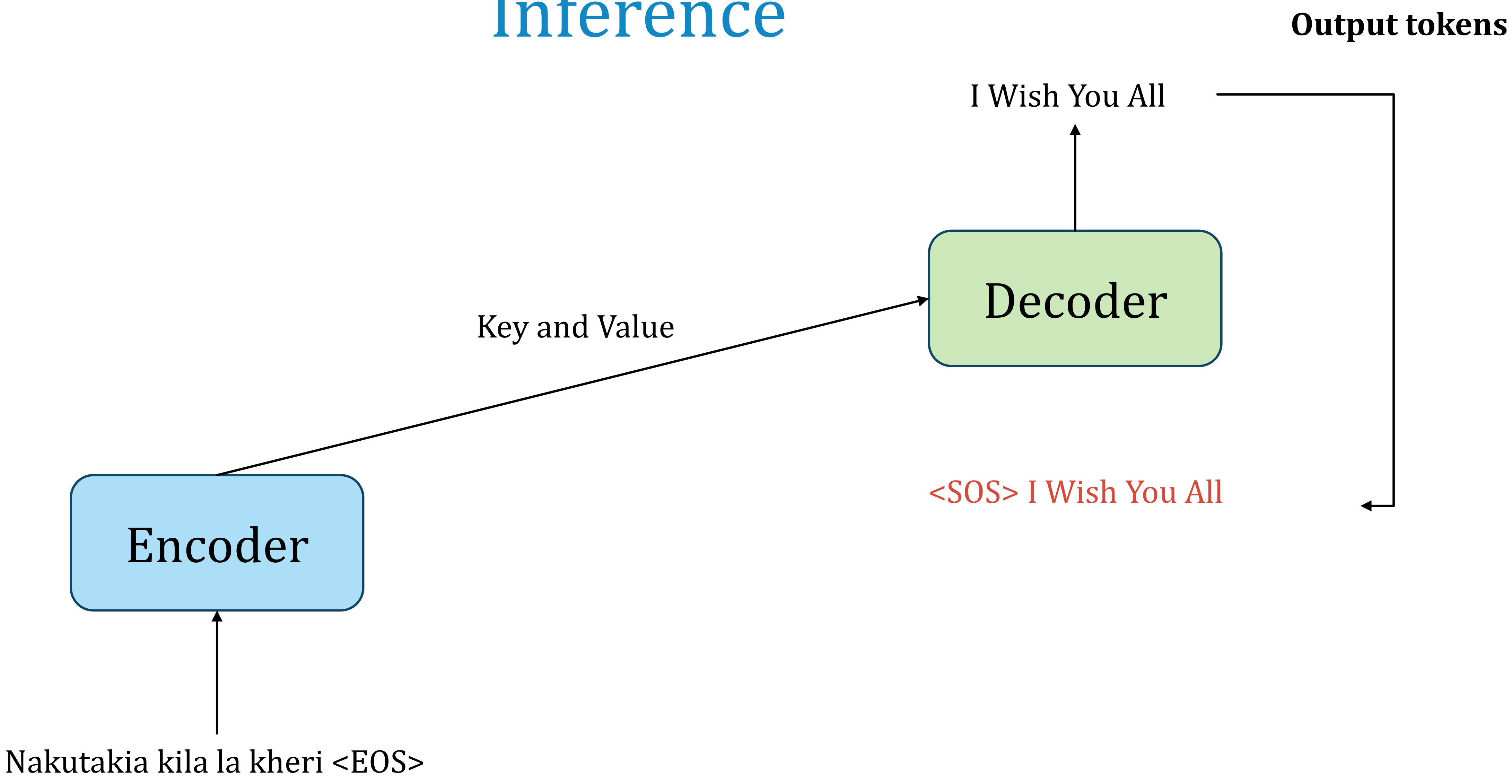

Output tokens
I Wish You
Decoder
Key and Value
<SOS> I Wish You
Encoder
Nakutakia kila la kheri <EOS>

Inference

Output tokens

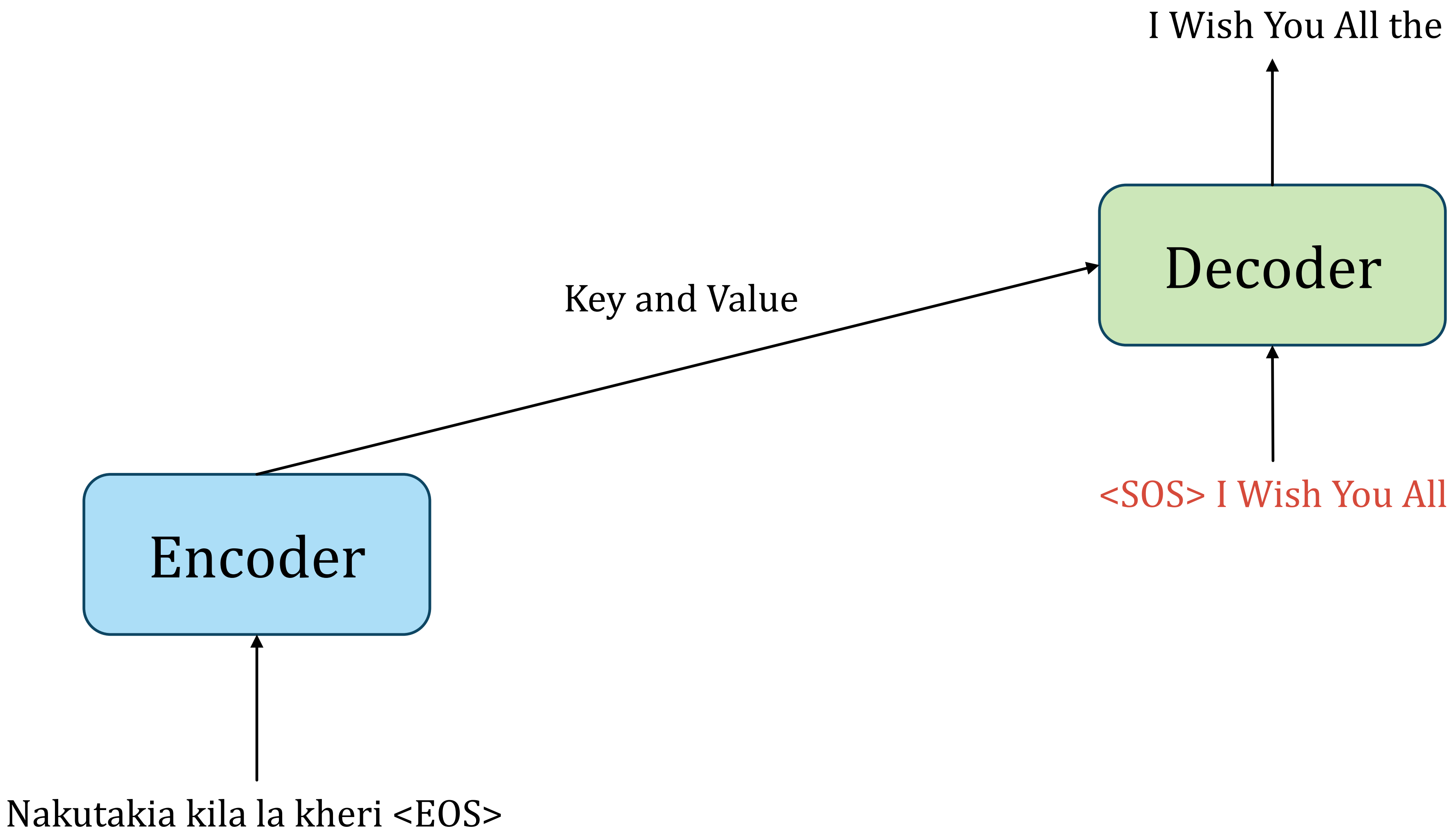


Inference

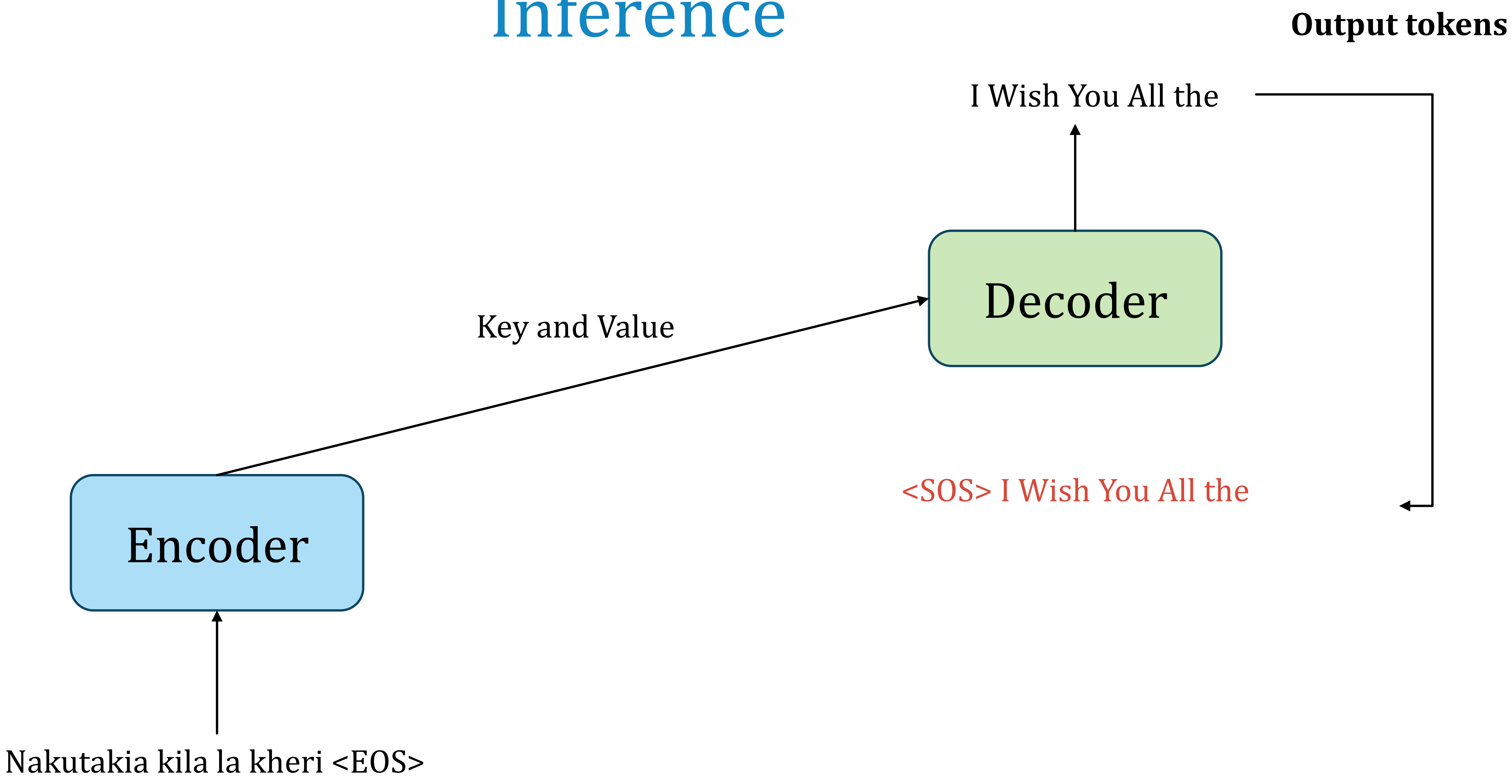

Output tokens
I Wish You All
Decoder
Key and Value
<SOS> I Wish You All
Encoder
Nakutakia kila la kheri <EOS>

Inference

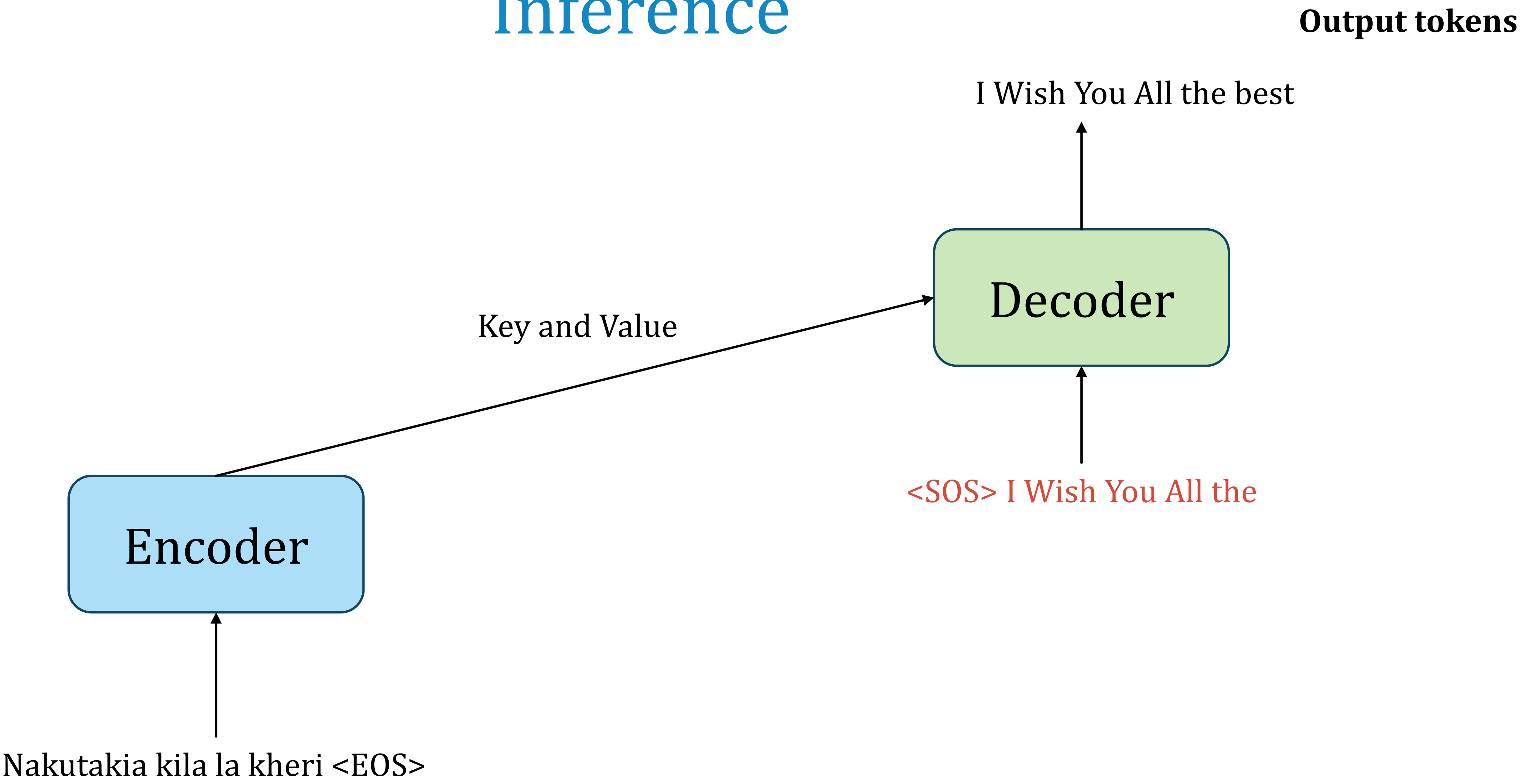
Output tokens



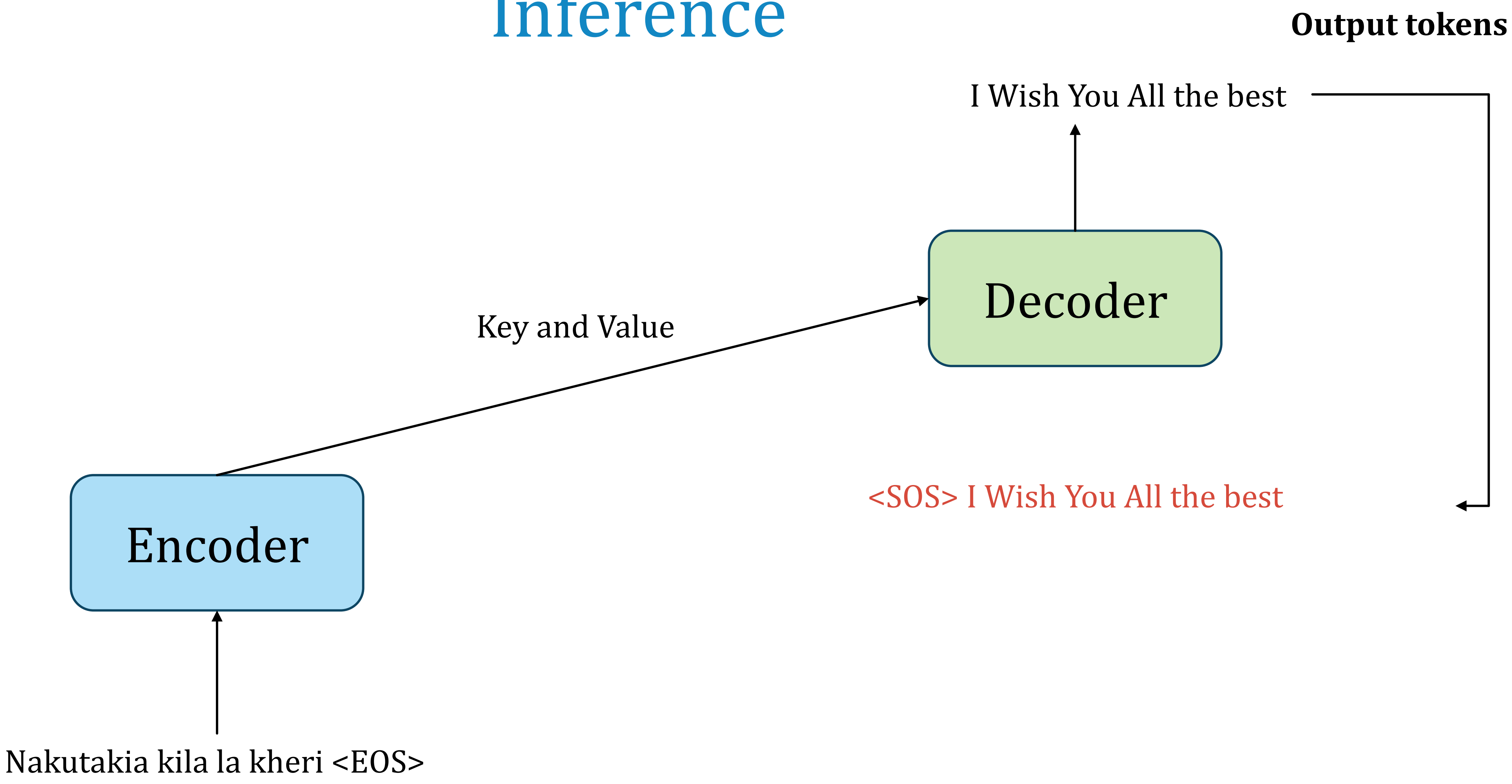
Inference


Output tokens
I Wish You All the
Decoder
Key and Value
<SOS> I Wish You All the
Encoder
Nakutakia kila la kheri <EOS>

Inference

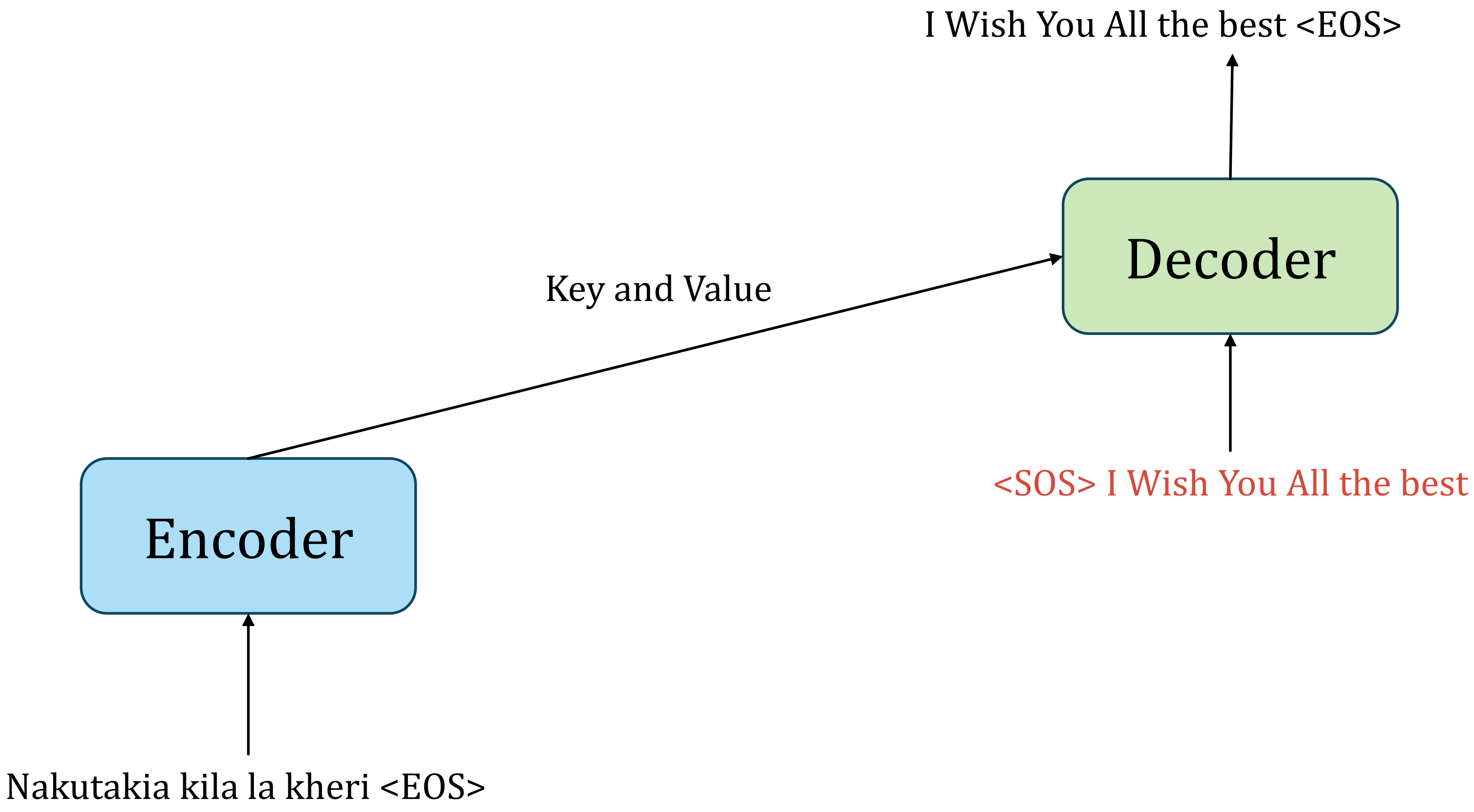

Output tokens
I Wish You All the best
Decoder
Key and Value
<SOS> I Wish You All the
Encoder
Nakutakia kila la kheri <EOS>

Inference

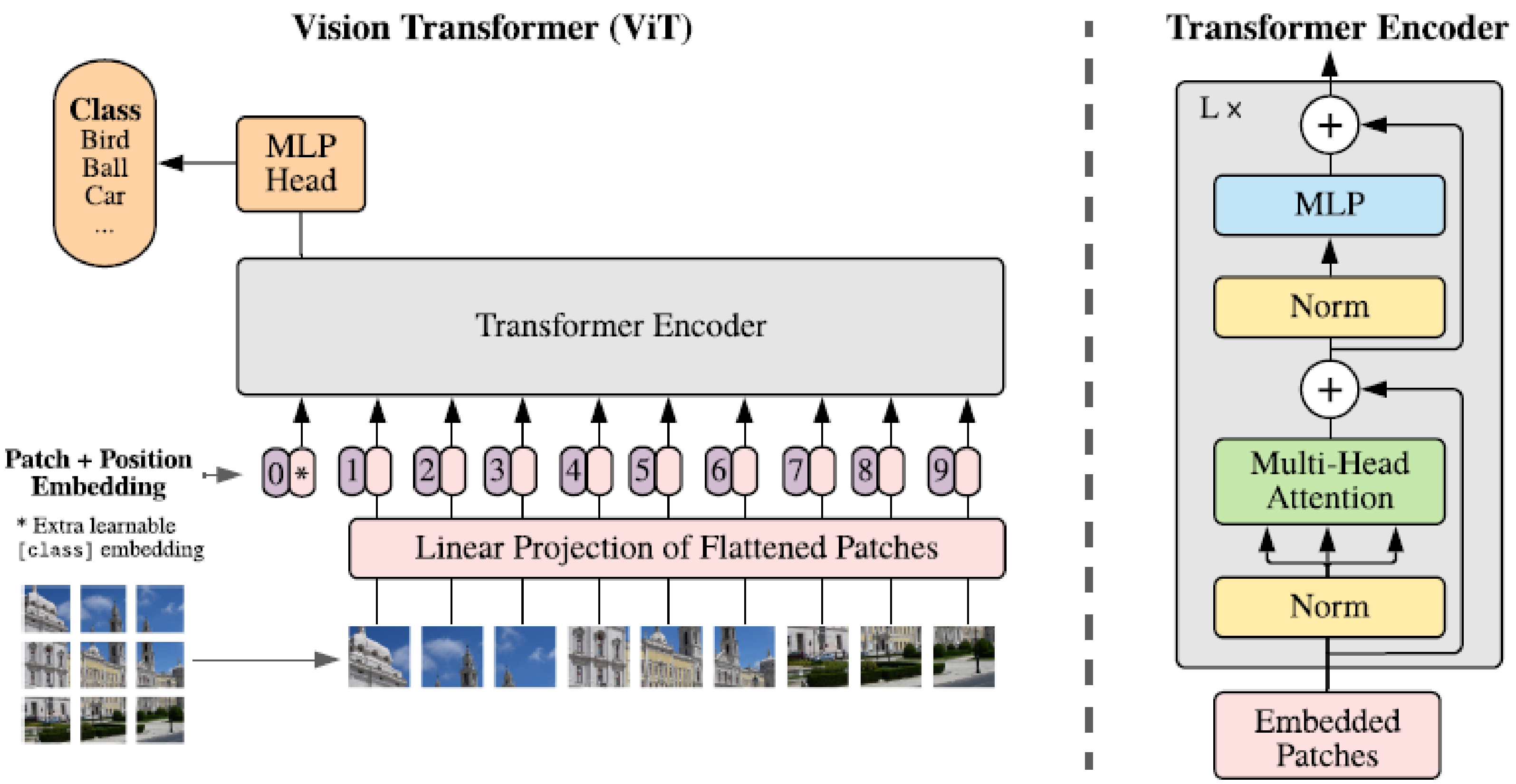

Output tokens
I Wish You All the best
Decoder
Key and Value
<SOS> I Wish You All the best
Encoder
Nakutakia kila la kheri <EOS>

Inference

Output tokens

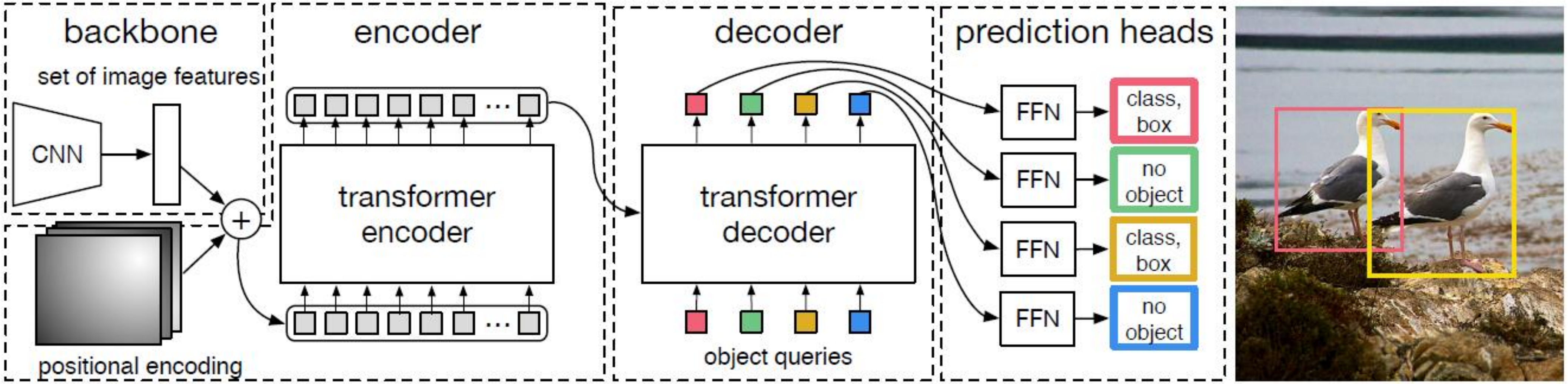


Vision Transformer



Dosovitskiy *et al.*, “AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE”. ICLR. 2021

Detection Transformer



Carion *et al.*, “End-to-End Object Detection with Transformers”. ECCV. 2021