# Transformer

Introduced at "Attention is all you need" by Ashish Vaswani et al. (2017)

## Story of attention

- sequence to sequence models [Sequence to Sequence Learning with Neural Networks, by Sutskever et. al. , 2014]

- attention mechanism [Neural Machine Translation by Jointly Learning to Align and Translate, Bahdanau et. al., 2015]

- self-attention [Long Short-Term Memory-Networks for Machine Reading, Cheng et. al., 2015]

## Transformer

- model architecture [Attention Is All You Need, Vaswani et. Al., 2017 ]

- multi-head attention layer

Task

• e.g. machine translation

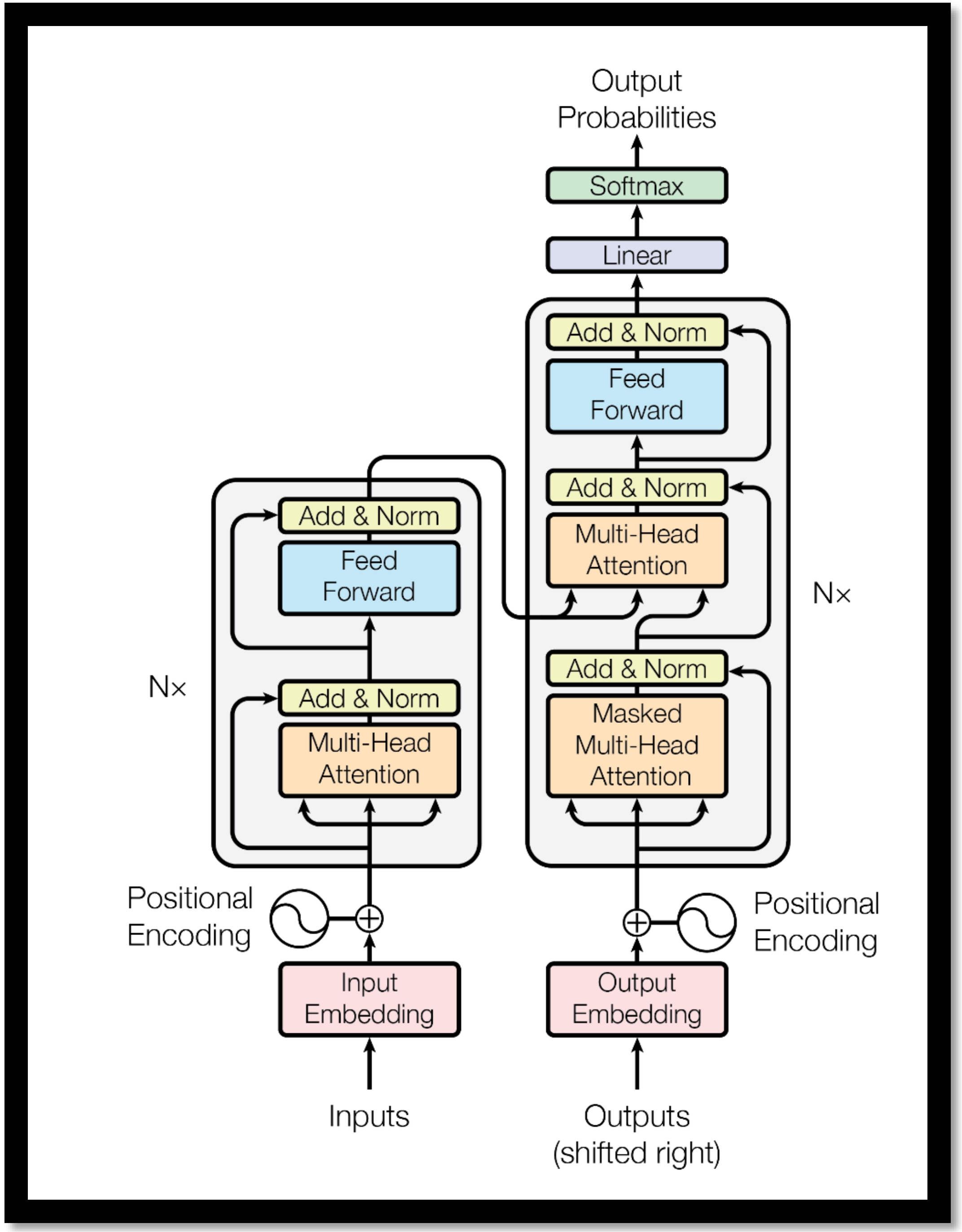
Input: source sequence
Output: target sequence

Architecture

* Feed-forward layers

* Attention layers

** No convolutional or recurrent layer **



# What is attention?

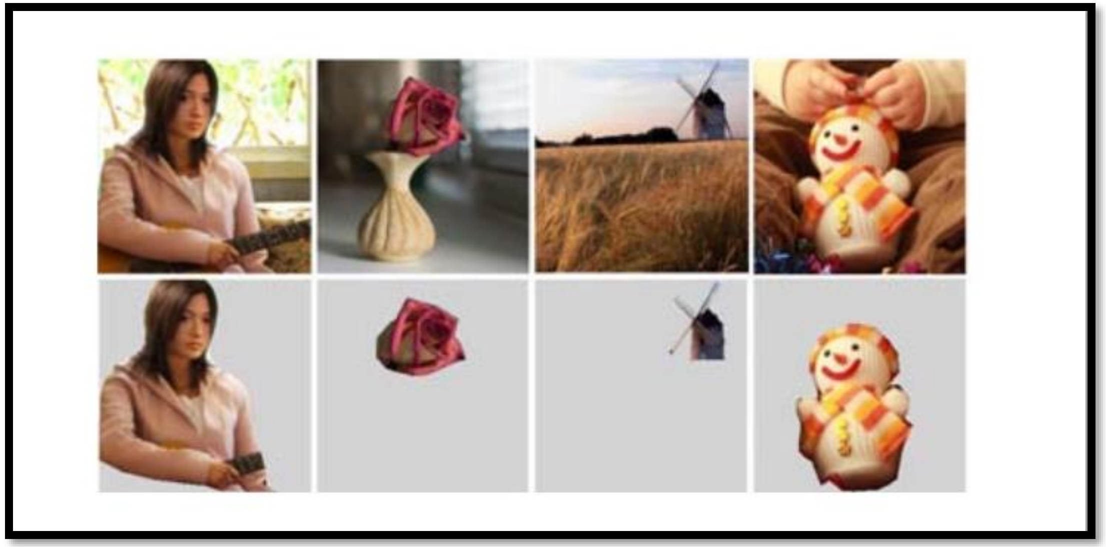
## Example 1

The girl who is playing with the ball is wearing a jacket.

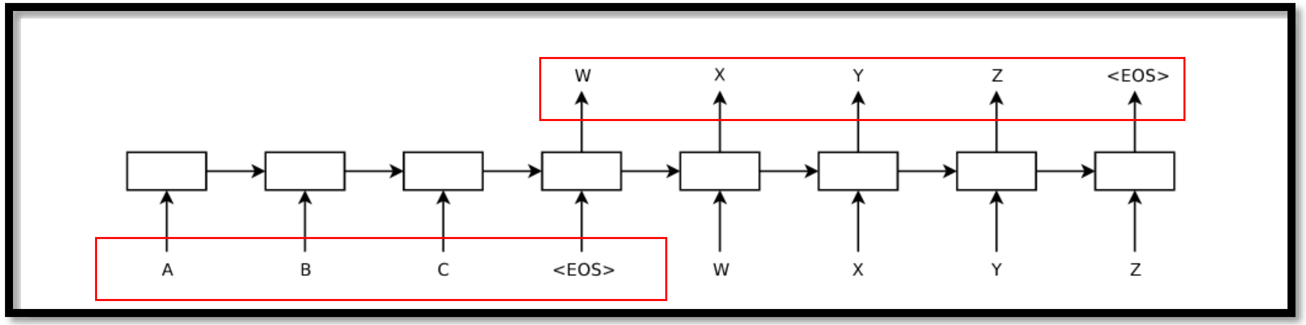
The girl who is playing with the $\boxed{\phantom{xx}}$ is wearing a jacket.

The girl who is playing with the ball is wearing a $\boxed{\phantom{xx}}$

## Example 2



# Story of attention mechanism: seq2seq modeling




Task:

Source sequence ⟶ target sequence

- Introduced for language modeling

Sequence to Sequence Learning with Neural Networks, by Sutskever et. al. , 2014

- Found application at:

* Machine translation (audio/text)
* QA dialogue generation
* Image caption generation

Method:

- Sequential modeling (RNN/CNN)
- Encoder-decoder architecture with fixed-length context vector

Limitations:

- No explicit mechanism for reasoning over structure (imposes an inductive bias to the structure of data).
- Not suitable for long sequences.

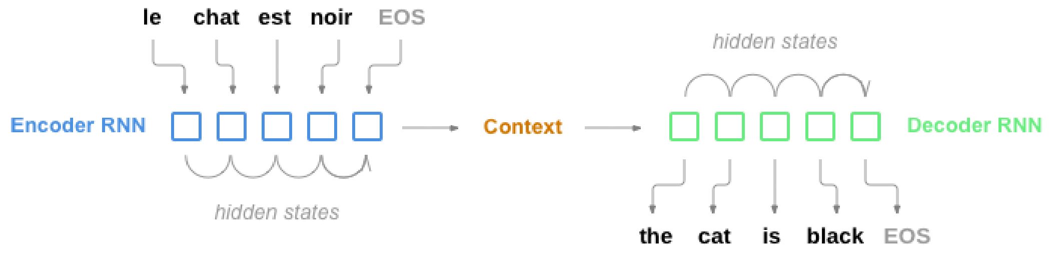
# Story of attention mechanism: seq2seq modeling

Encoder :

maps input sequence
to a context vector

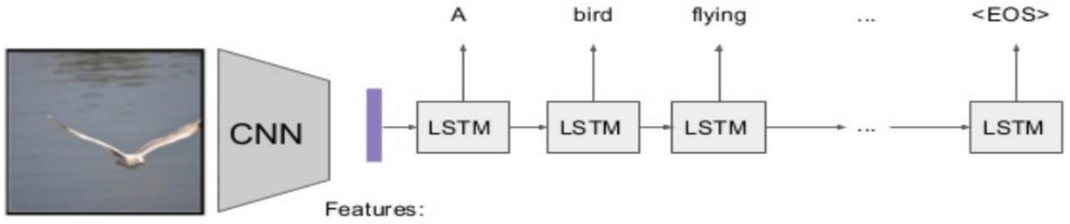
Encoder input: variable-length source sentence

Decoder output: variable-length target sentence




Decoder :

maps the context vector
to another sequence

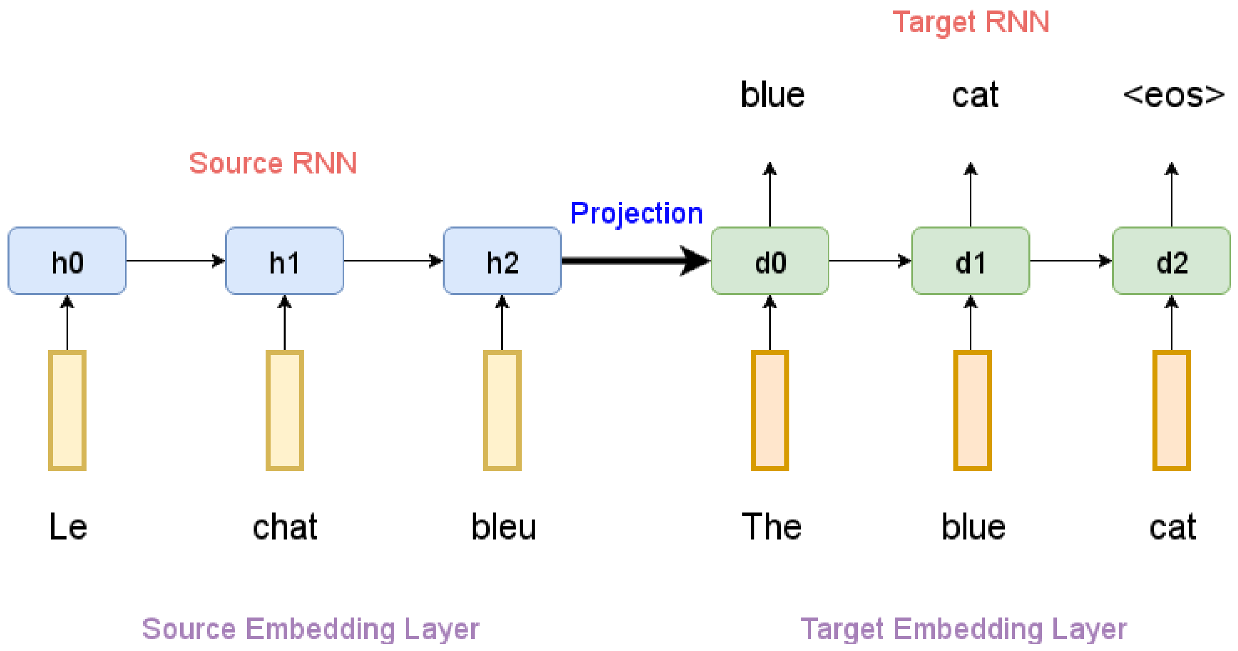



The network compresses all source information into a static fixed-length context vector
And thus all output predictions are based on static output of encoder.

# Story of attention mechanism: seq2seq modeling

$$h_t = \mathrm{sigm}\left(W^{\mathrm{hx}} x_t + W^{\mathrm{hh}} h_{t-1}\right)$$
$$y_t = W^{\mathrm{yh}} h_t$$

initial hidden state is set to the representation $v$ of $x_1, \ldots, x_T$




The model tries to estimate conditional probability of:

$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

# Story of attention mechanism:

# attention layer

- Introduced for NMT :

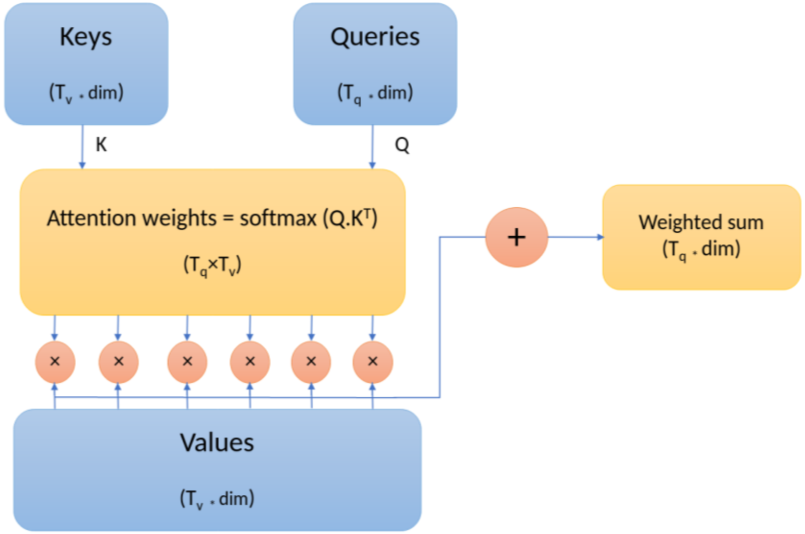
To automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

Neural Machine Translation by Jointly Learning to Align and Translate, Bahdanau et. al., 2015

Advantages:

- The model automatically finds correspondence between source and target sequences (alignment)

- Suitable for long sequences




Attention layer returns an output based on input query and its memory.

# Story of attention mechanism:

## attention layer

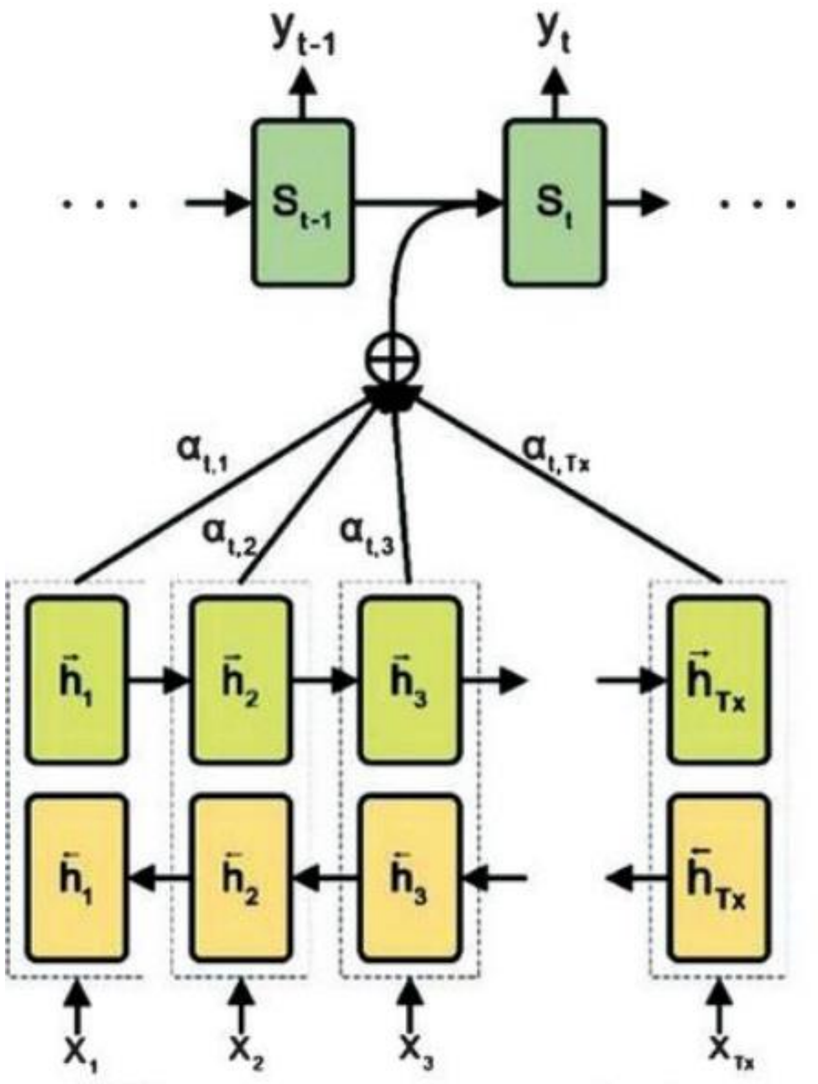
$$p(y_i|y_1,\ldots,y_{i-1},\mathbf{x}) = g(y_{i-1},s_i,c_i),$$

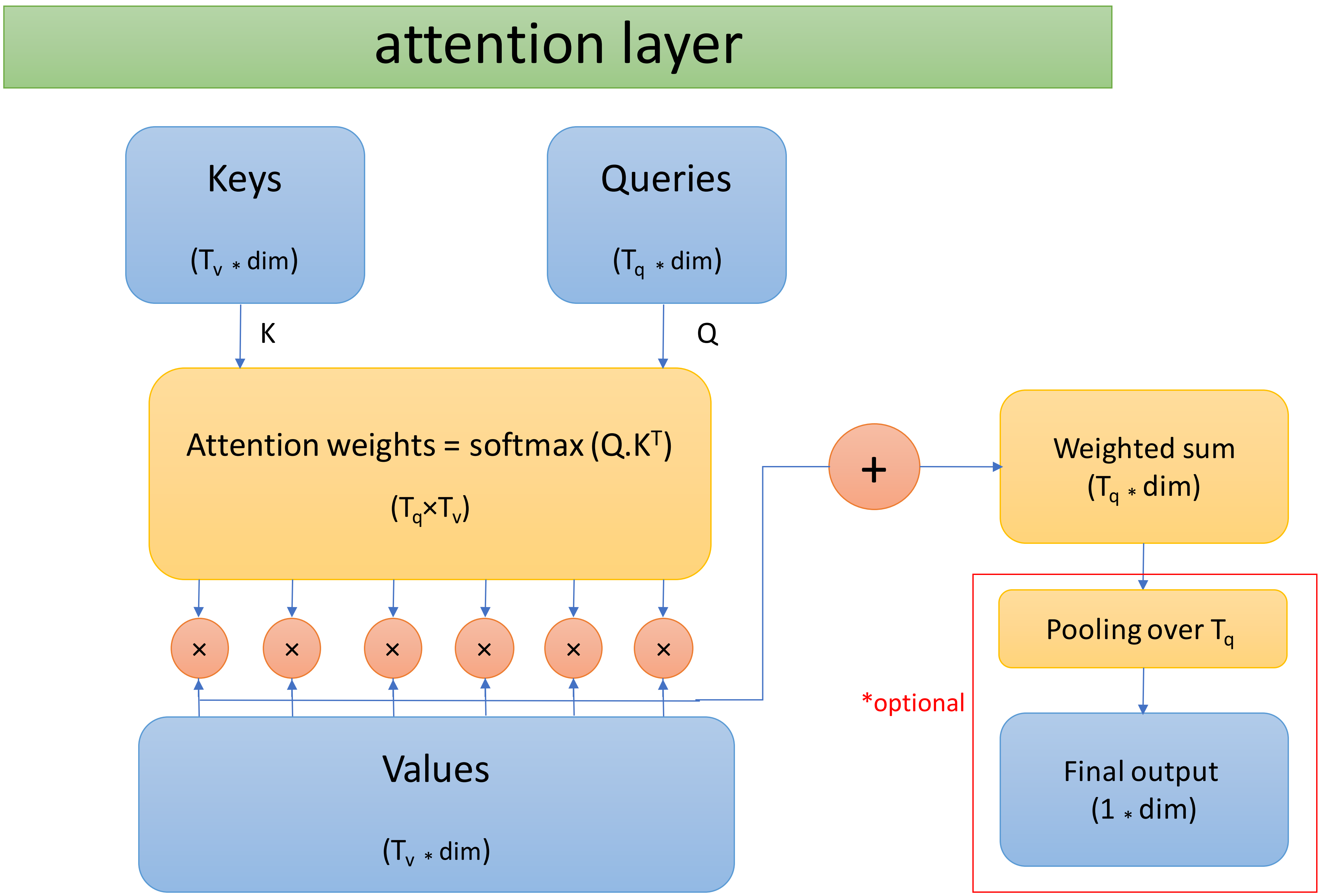
$$s_i = f(s_{i-1},y_{i-1},c_i)$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

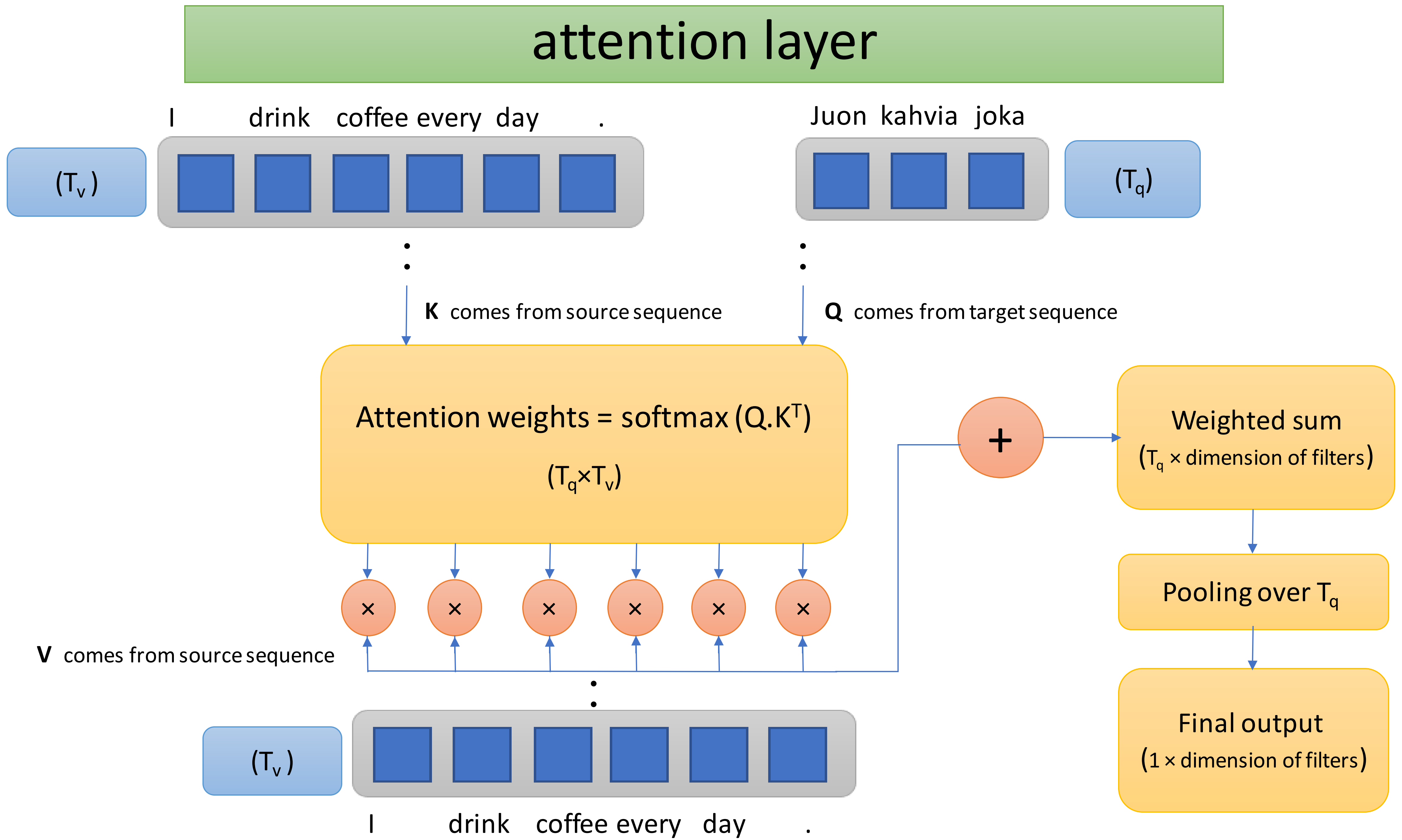
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x}\exp(e_{ik})},$$

$$e_{ij} = a(s_{i-1},h_j)$$



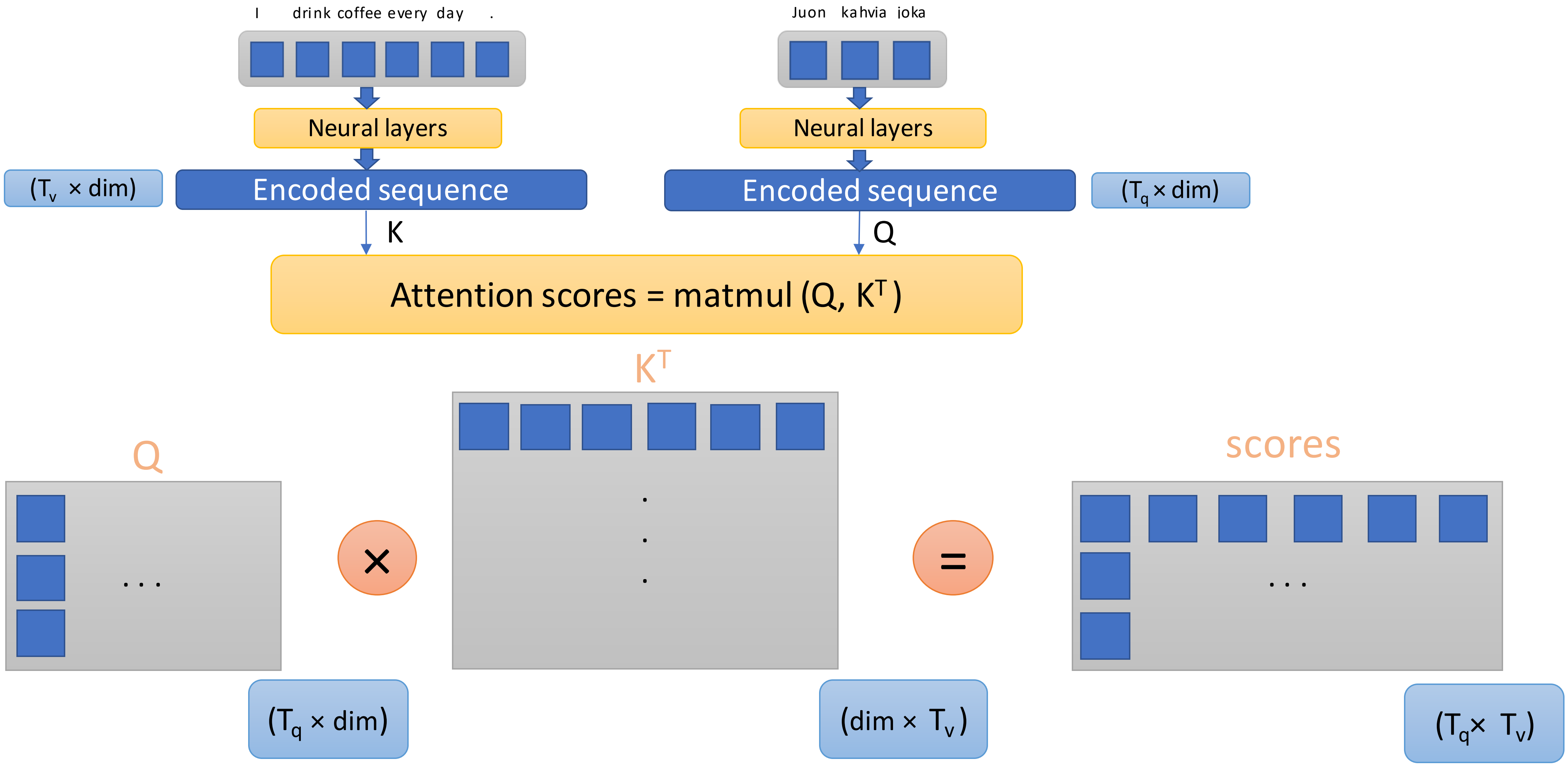


attention layer
Keys
(T_v * dim)
Queries
(T_q * dim)
K
Q
Attention weights = softmax (Q.K^T)
(T_q×T_v)
+
Weighted sum
(T_q * dim)
Pooling over T_q
*optional
Final output
(1 * dim)
Values
(T_v * dim)

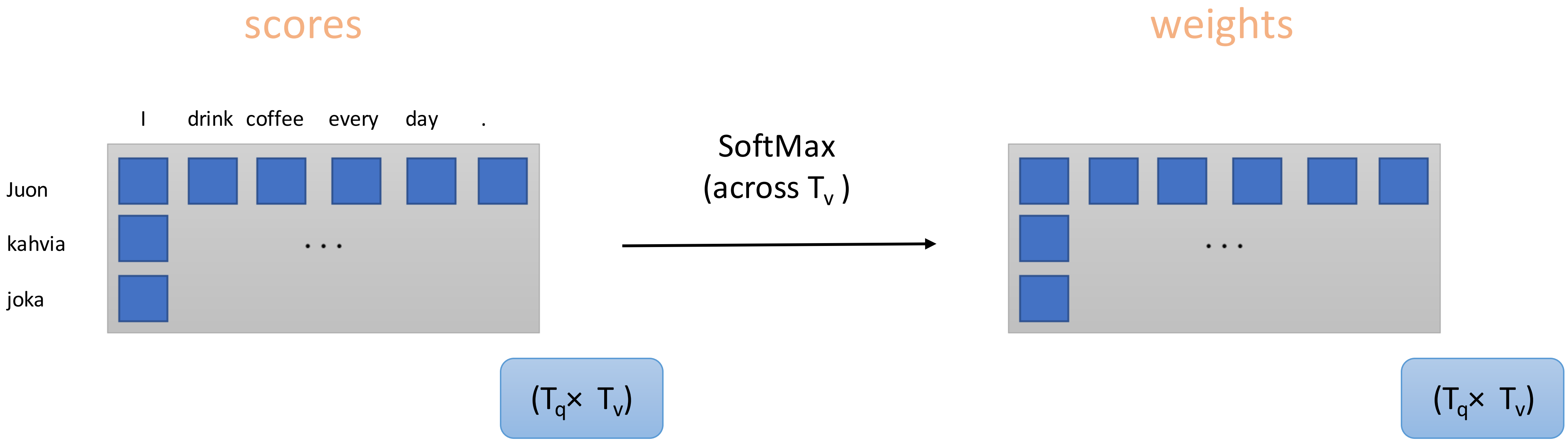


attention layer
I drink coffee every day .
$(T_v)$
Juon kahvia joka
$(T_q)$
K comes from source sequence
Q comes from target sequence
Attention weights = softmax $(Q.K^T)$
$(T_q \times T_v)$
Weighted sum
$(T_q \times$ dimension of filters$)$
Pooling over $T_q$
Final output
$(1 \times$ dimension of filters$)$
V comes from source sequence
$(T_v)$
I drink coffee every day .

# 1 attention scores



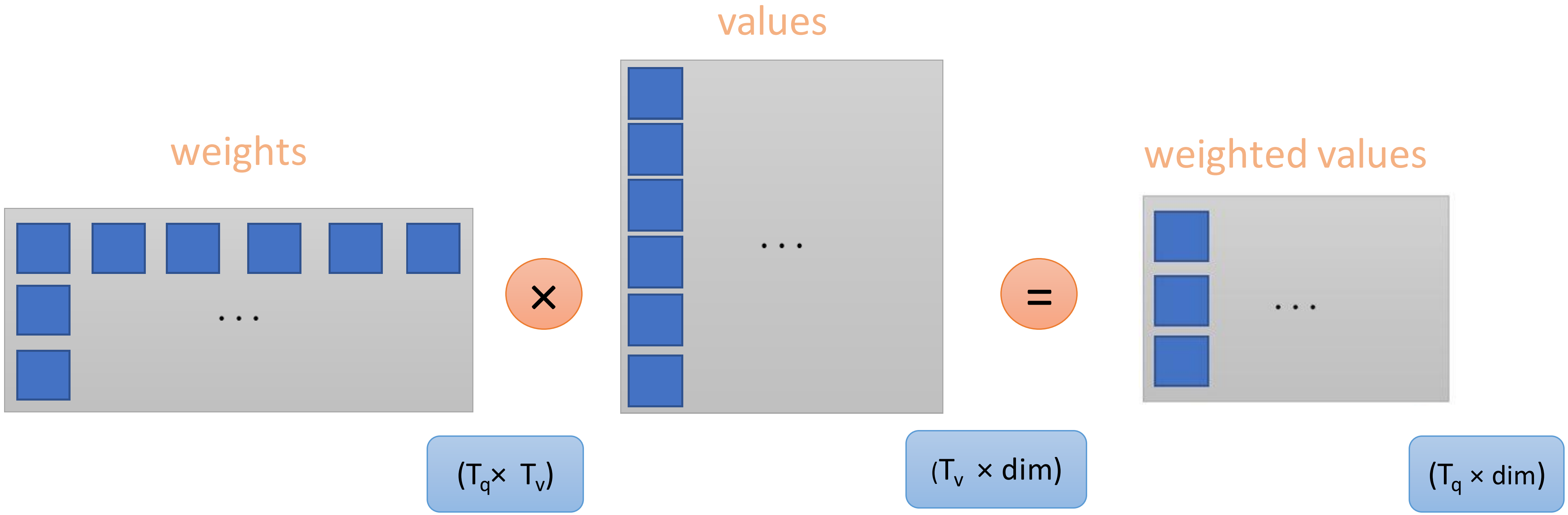
# 2 attention weights



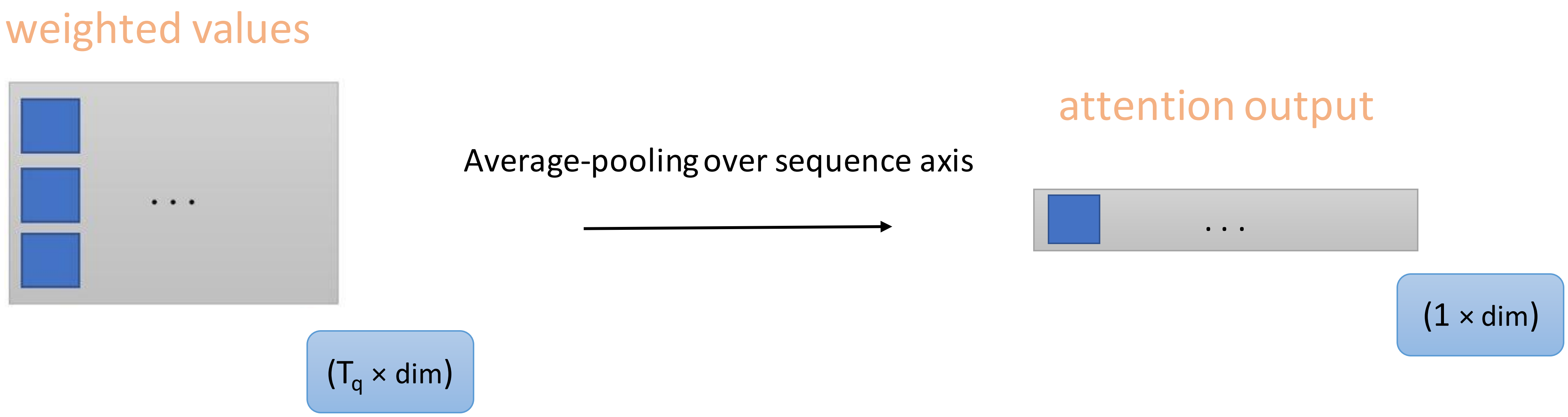

SoftMax = exp(logits) / reduce_sum(exp(logits), axis=-1)

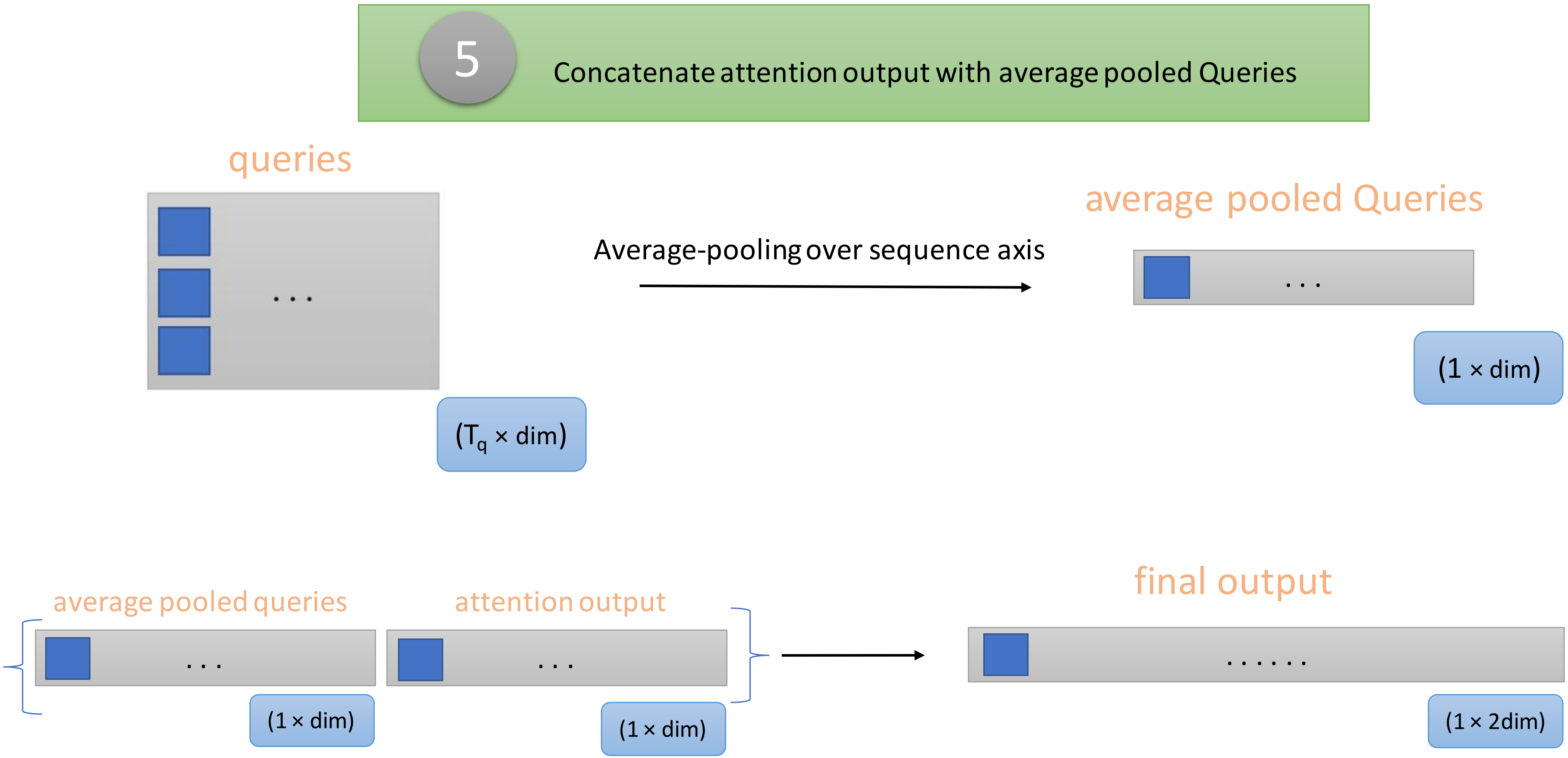
For each query instance (e.g. kahvia) we have a distribution of weights over values.

3 Using weights to create a linear combination of Values

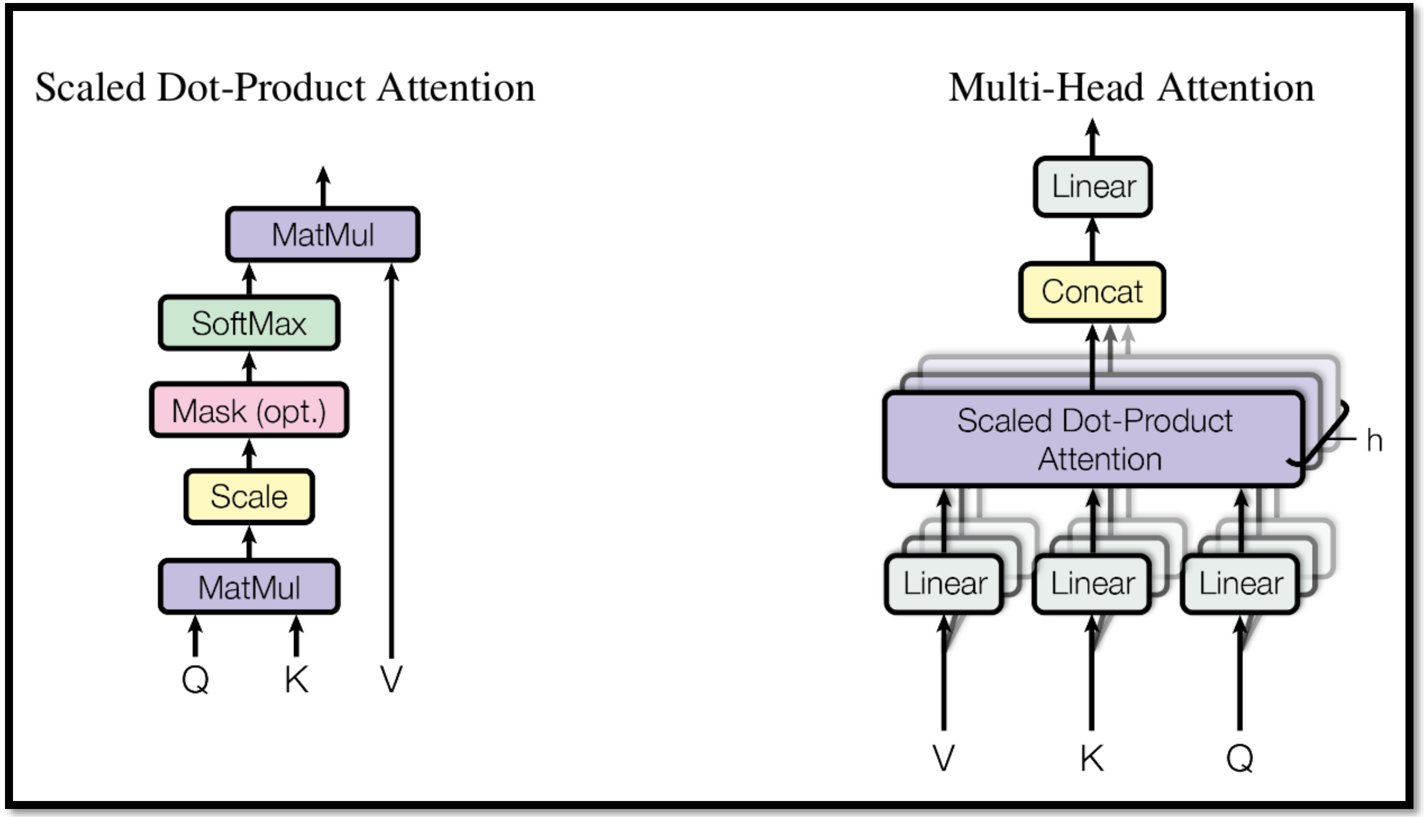


# 4 average pooling



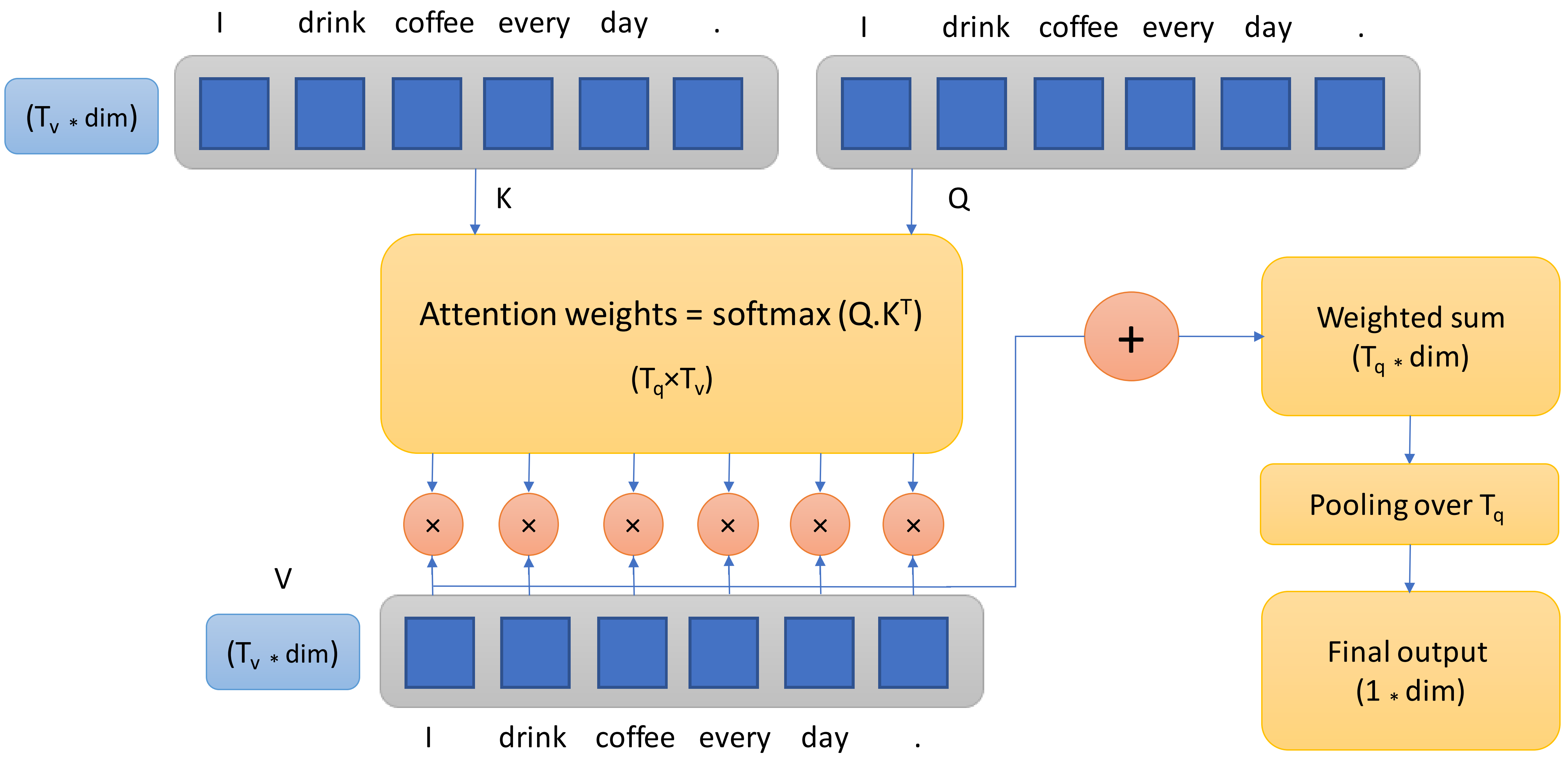

5
Concatenate attention output with average pooled Queries
queries
(T$_q$ × dim)
Average-pooling over sequence axis
average pooled Queries
(1 × dim)
average pooled queries
(1 × dim)
attention output
(1 × dim)
final output
(1 × 2dim)

# attention layer




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

# Self-attention layer



# Story of attention mechanism: Self-attention

- Introduced for "Machine Reader" (understanding the text):

Offers a way to weakly induce relations among tokens by modifing the standard LSTM structure by replacing the memory cell with a memory network (LSTMN).

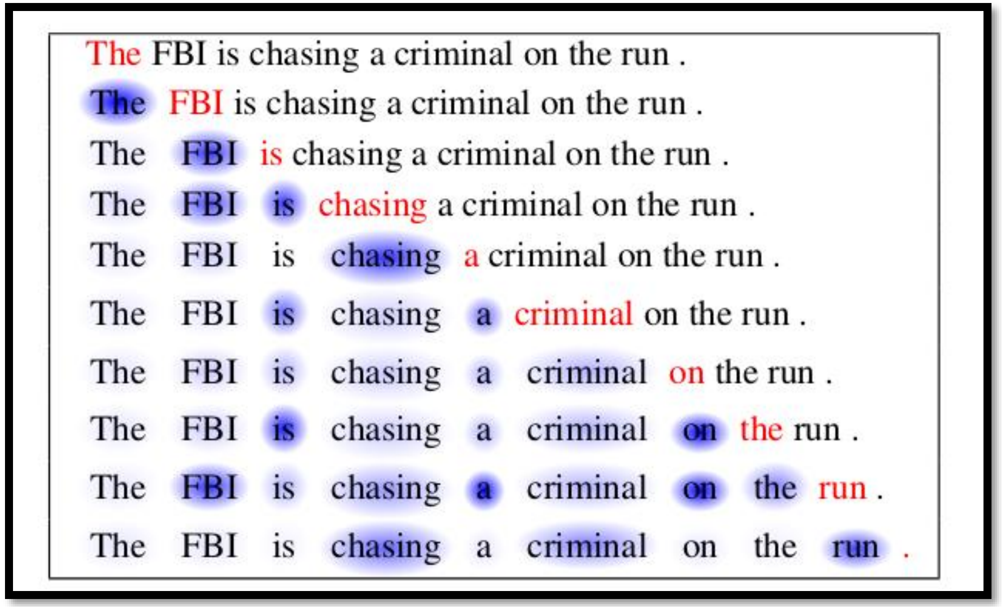
Long Short-Term Memory-Networks for Machine Reading, Cheng et. al., 2015

- Found application at:

* language modeling
* sentiment analysis
* ?

Red represents current token

Blue represents memories

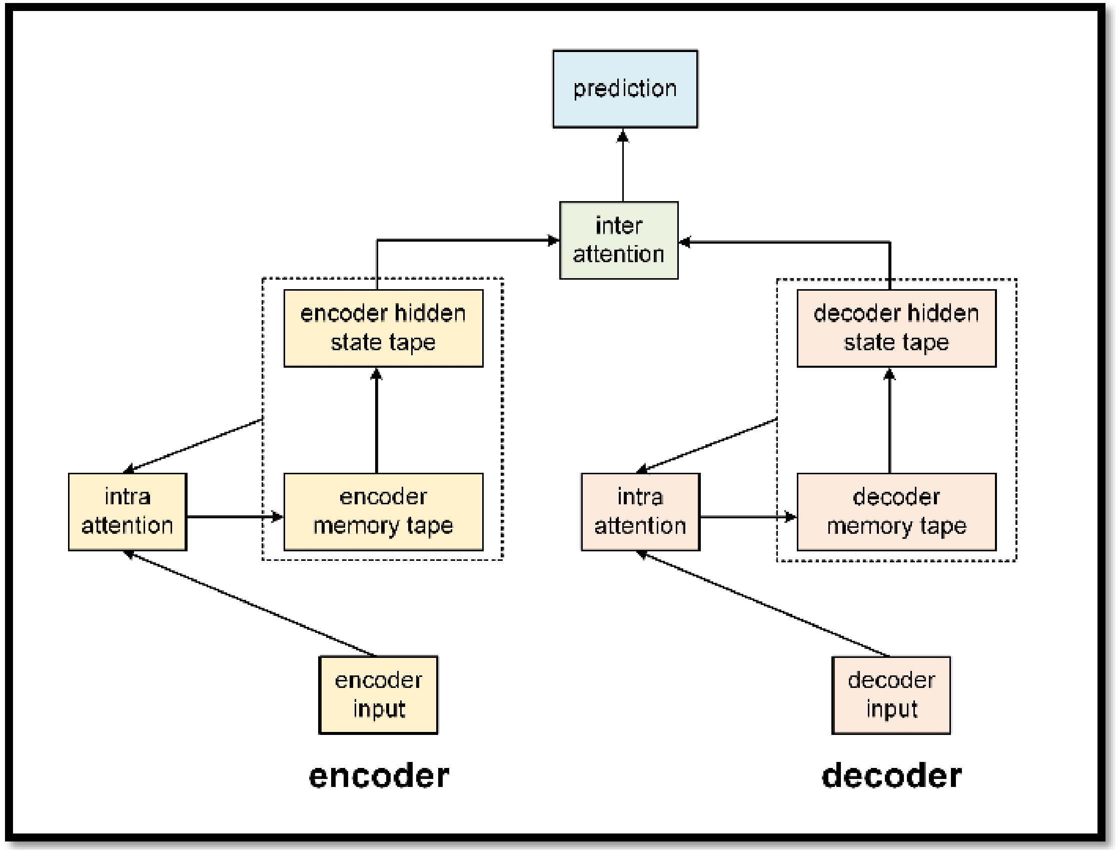



- Text is processes incrementally while learning which past tokens and to what extend they relate to the current token.

- Feature: the model induces undirected relations among tokens

- Method: attention is added within a sequence encoder

# Story of attention mechanism: Self-attention




Shallow attention fusion, Cheng et. al, 2015

Self-attention, (also called intra-attention), is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence.

# Story of attention mechanism: Aligning functions

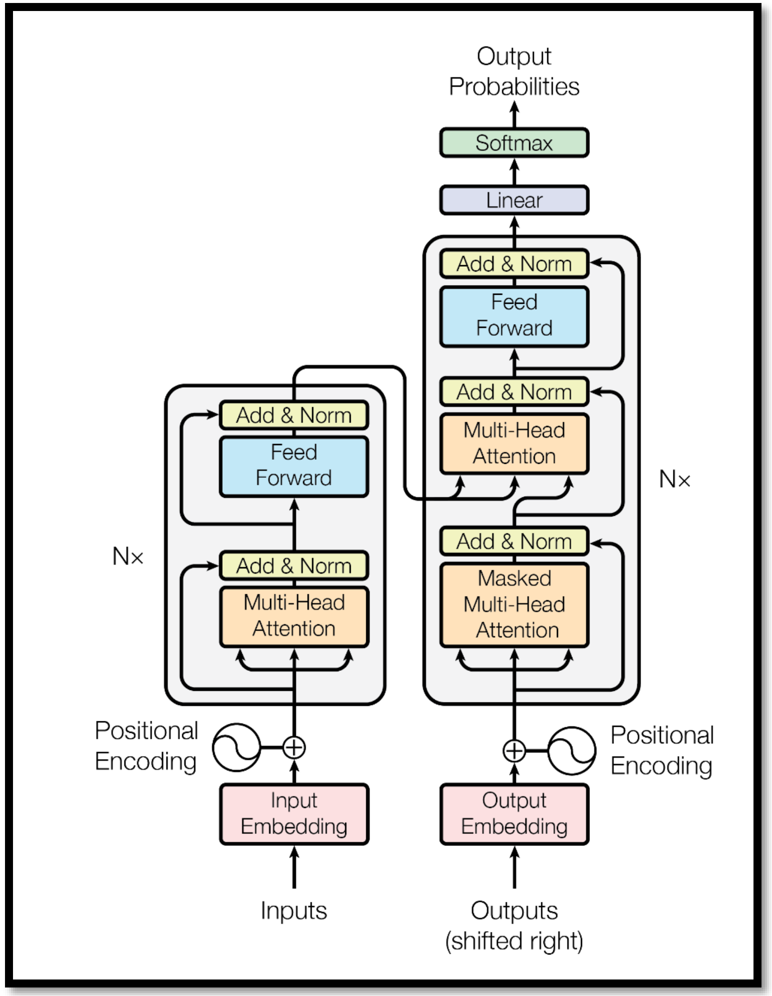
```
scores = tf.reduce_sum(tf.tanh(query  + value), axis=-1)
```

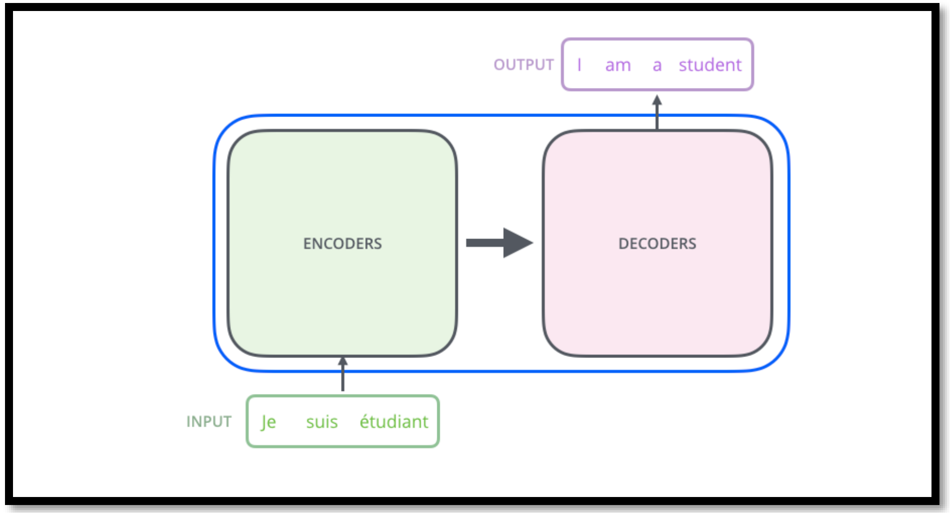
```
scores = tf.matmul(query,  key, transpose_b=True)
```

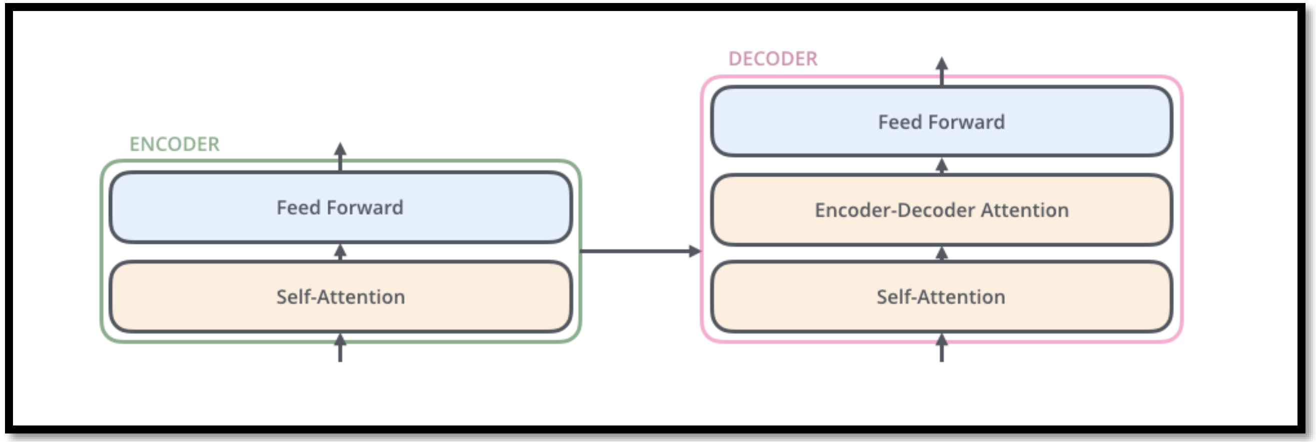
| Name | Alignment score function | Citation |
|---|---|---|
| Content-base attention | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$ | Graves2014 |
| Additive(*) | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$ | Bahdanau2015 |
| Location-Base | $\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$<br>Note: This simplifies the softmax alignment to only depend on the target position. | Luong2015 |
| General | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$<br>where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer. | Luong2015 |
| Dot-Product | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$ | Luong2015 |
| Scaled Dot-Product(^) | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$<br>Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state. | Vaswani2017 |

# Transformer

# (multi-head self-attention)


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
I am a student
ENCODERS
DECODERS
INPUT
Je suis étudiant

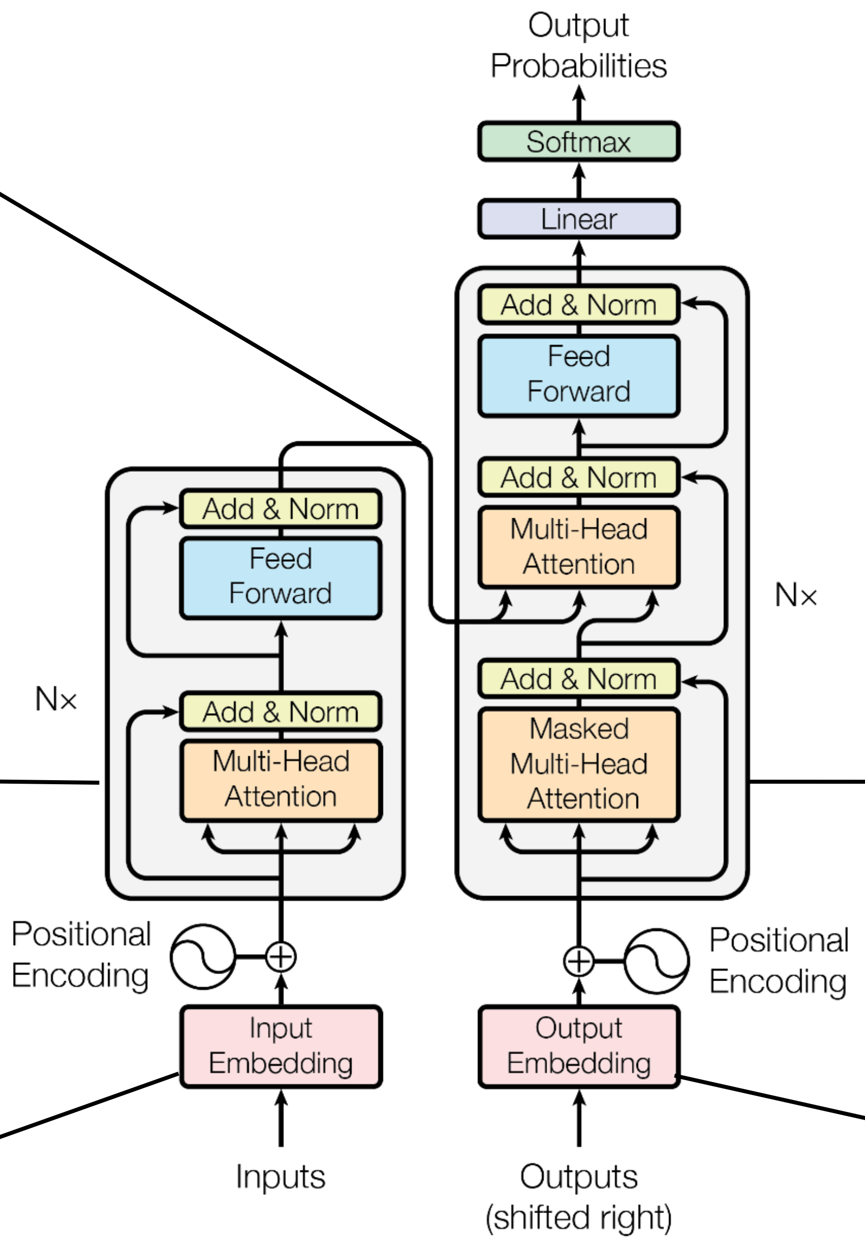

ENCODER
Feed Forward
Self-Attention
DECODER
Feed Forward
Encoder-Decoder Attention
Self-Attention

encoder-decoder attention

Relying entirely on self-attention to compute representations of its input and output!

Self-attention: learning long-range dependencies

constantly updated during training



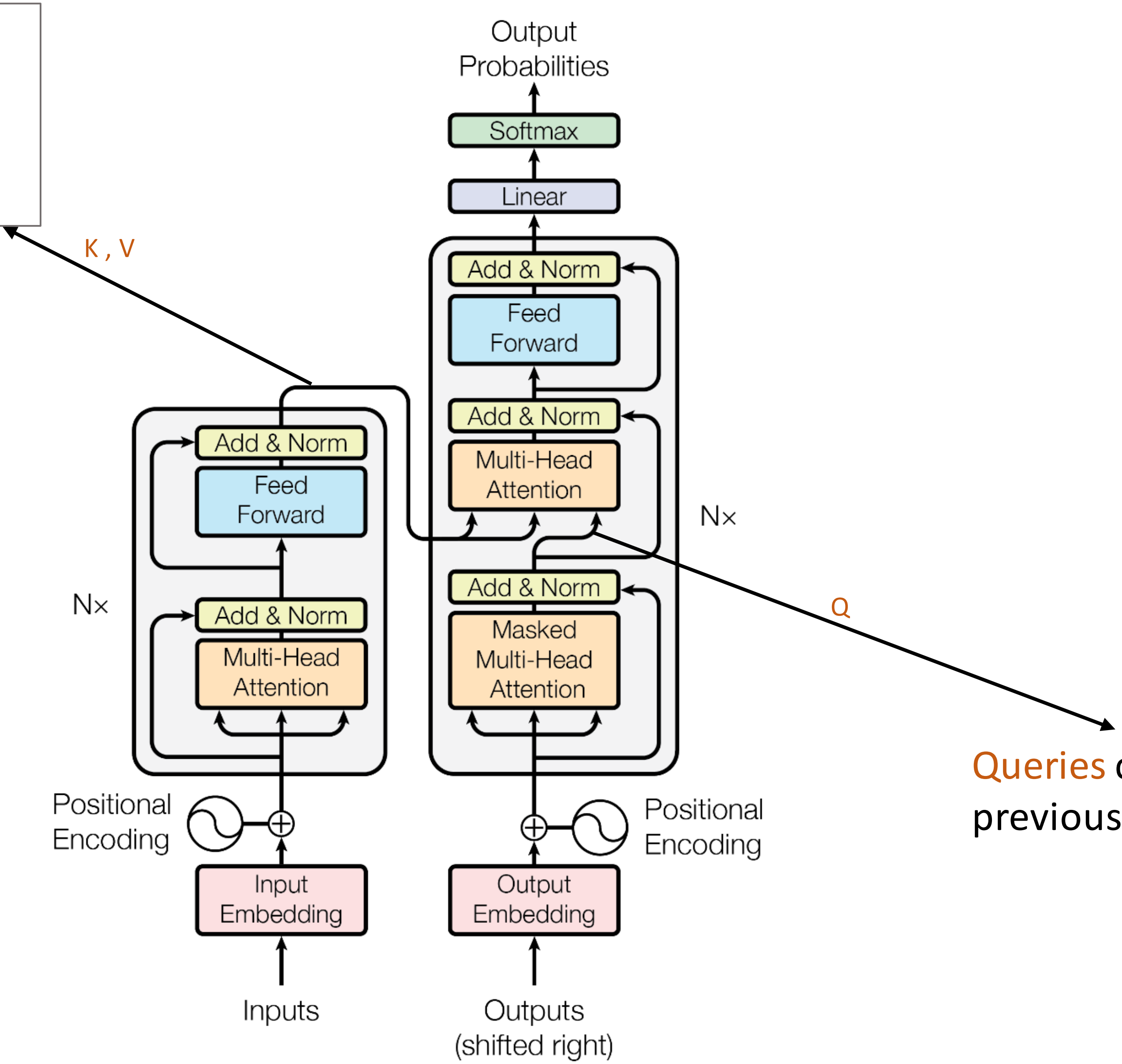

Auto-regressive : at each time step the model is consuming the previously generated symbols as additional input!

Self-attention: learning long-range dependencies

constantly updated during training

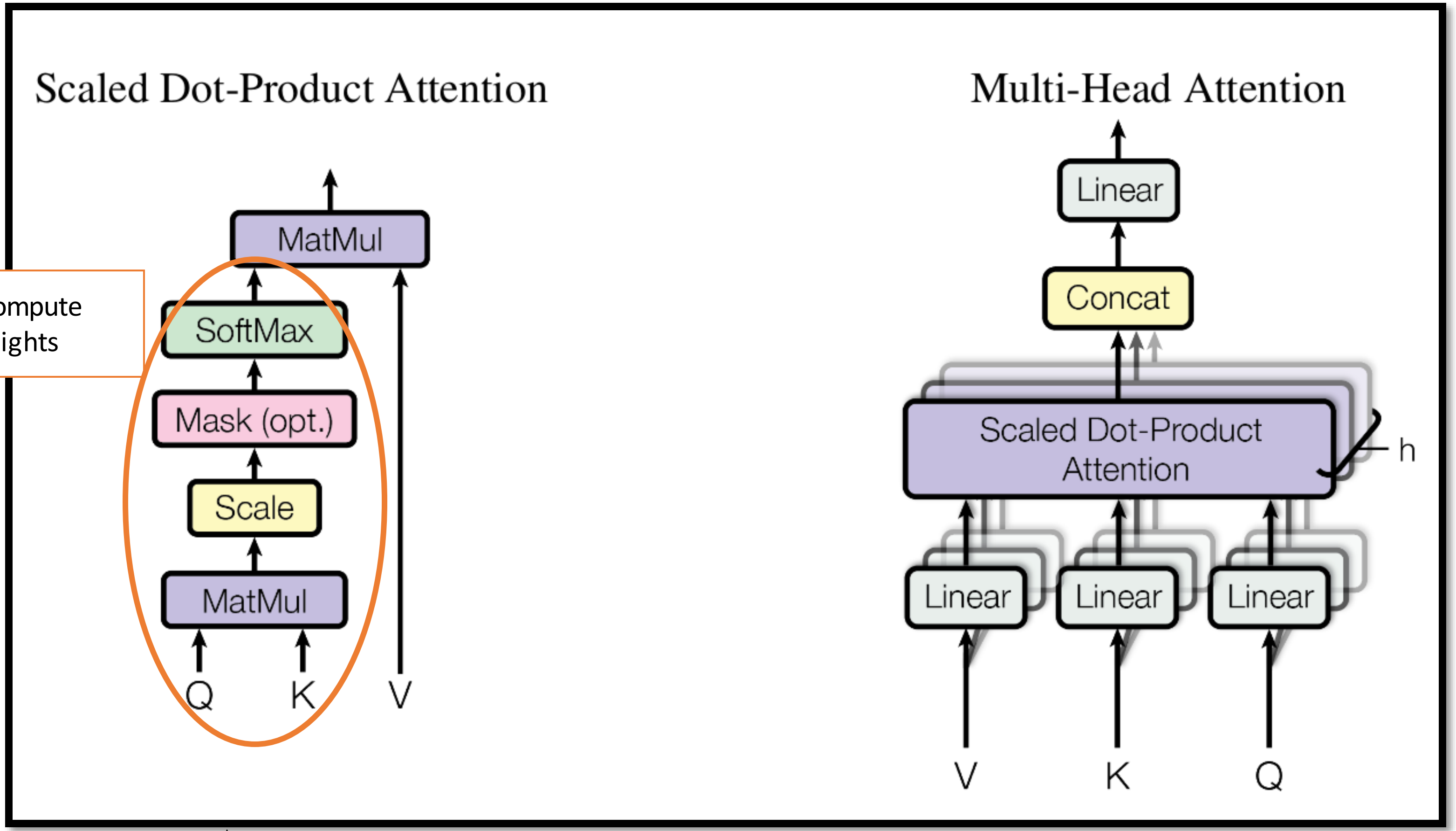
Memory Keys and Values

encoded representation of input
(e.g. encoder hidden states)




Queries come from the
previous decoder layer

To compute weights




back to d (model) = 512

8 parallel channels of size 64

- attention function is performed in each parallel channel separately
- allows the model to jointly attend to information from different subsets
- works better than averaging
- total computation cost is smaller

d (model) = 512

Weighted sum of values

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

## Summary

- Attention helps to focus on relatively important part of sequential data.
- Self-attention is an attention mechanism that relates different parts of a single sequence in order to make a representation of that sequence.
- Transformer relies entirely on self-attention to compute representation of input and output of the network without using  recurrent or convolution layers.

# Thank you!