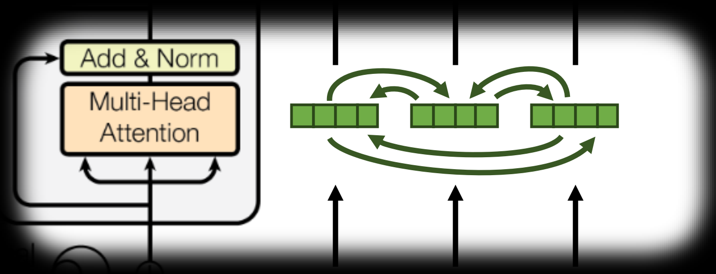
Neural Computation

Attention

Thanks to Constantin Pape and Alex Ecker!



Attention

https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

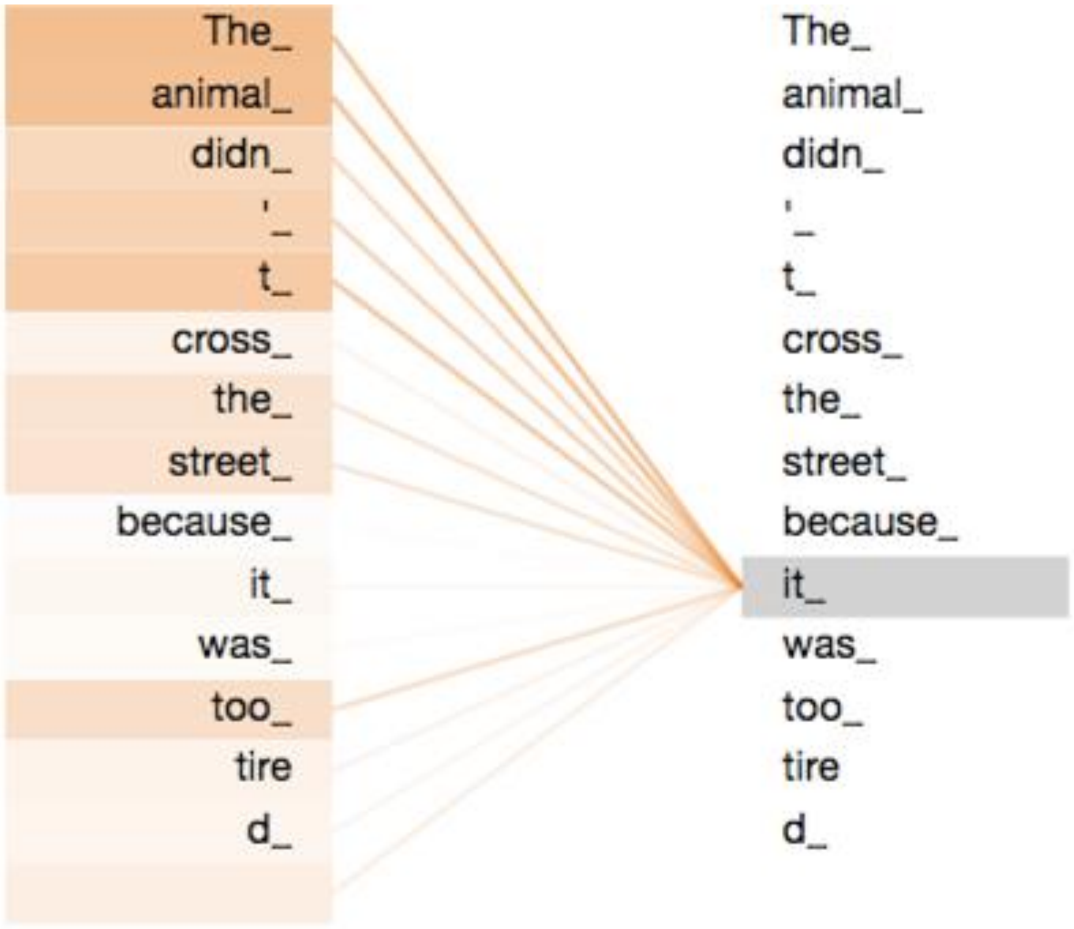
Self-Attention

Self-attention:

- every element in sequence can influence every other element
- learn weighting (“attention”) for each pair of elements

Attention: learnable pairwise weighting that depends on other sequence

Self-Attention: use input sequence for attention



Key-Value Pairs

JSON-Files:

```json
{
  "first_name": "John",
  "last_name": "Smith",
  "is_alive": true,
  "age": 27,
  "address": {
    "street_address": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postal_code": "10021-3100"
  },
  "phone_numbers": [
    {
      "type": "home",
      "number": "212 555-1234"
```

Python Dictionary:

```python
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

Key-Value Pairs

Query
Date of birth

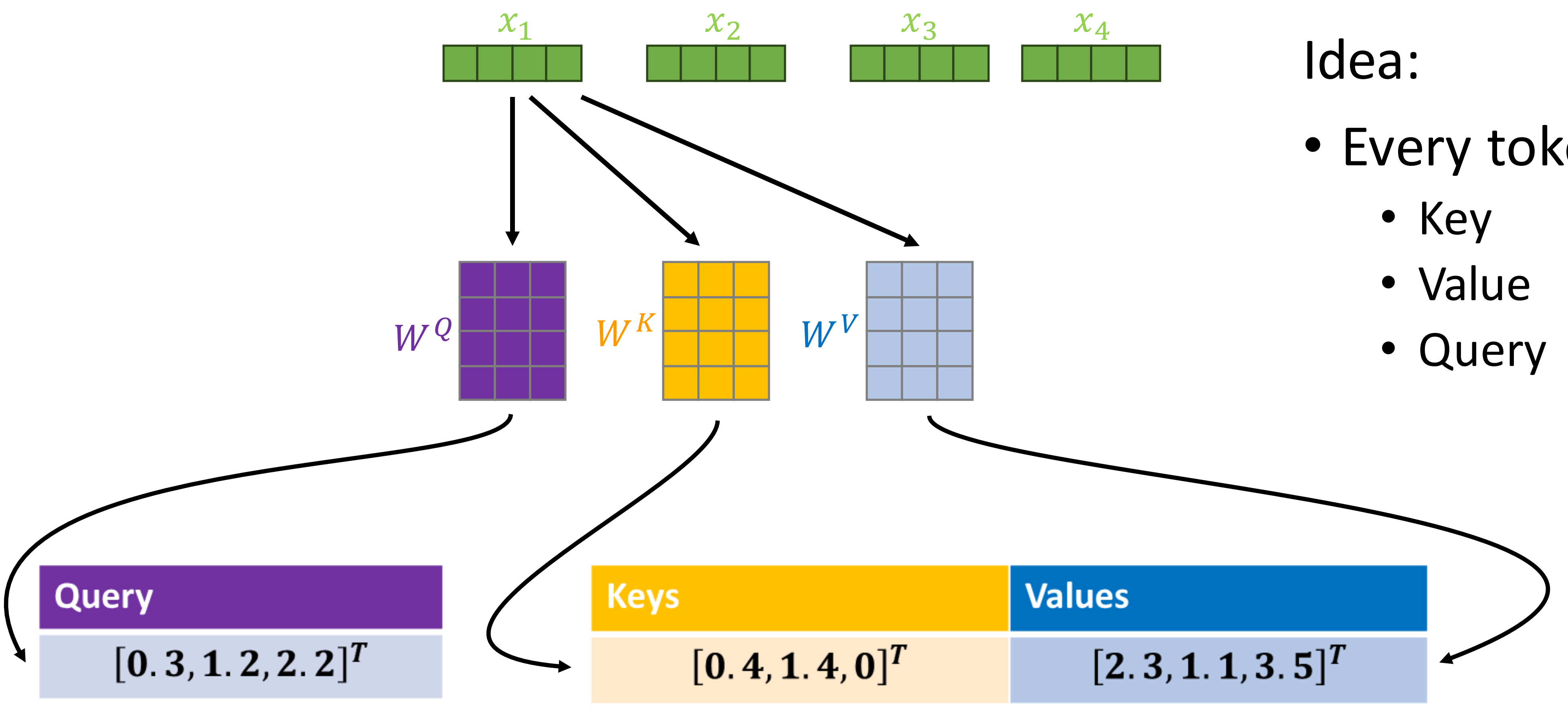
Keys	Values
Name	Jane Doe
Address	37 Coronation street B12 9TK
Date of birth	May 5th 2000
Place of birth	Hull

Key-Value Pairs

Query
Date of birth

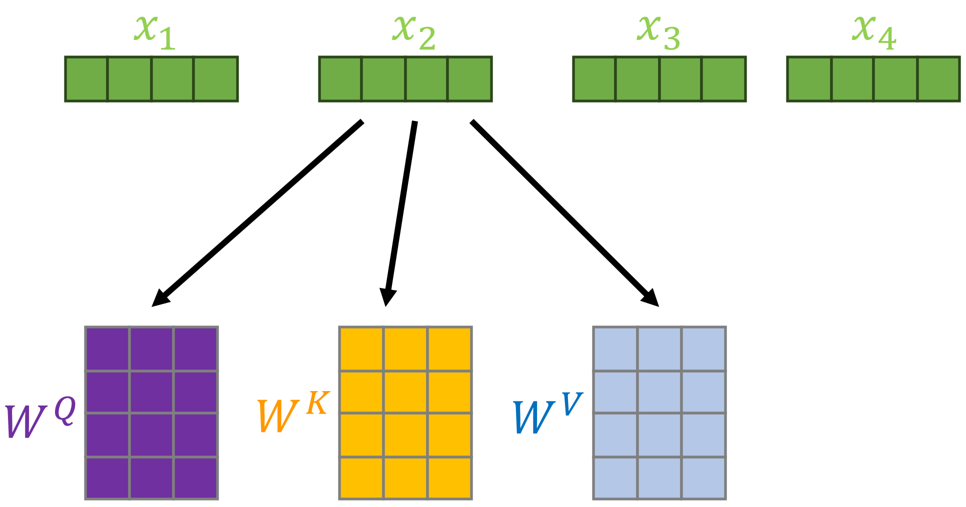
Keys	Values
Name	Jane Doe
Address	37 Coronation street B12 9TK
Date of birth	May 5th 2000
Place of birth	Hull

Result
May 5th 2000



Idea:

- Every token makes:
  - Key
  - Value
  - Query

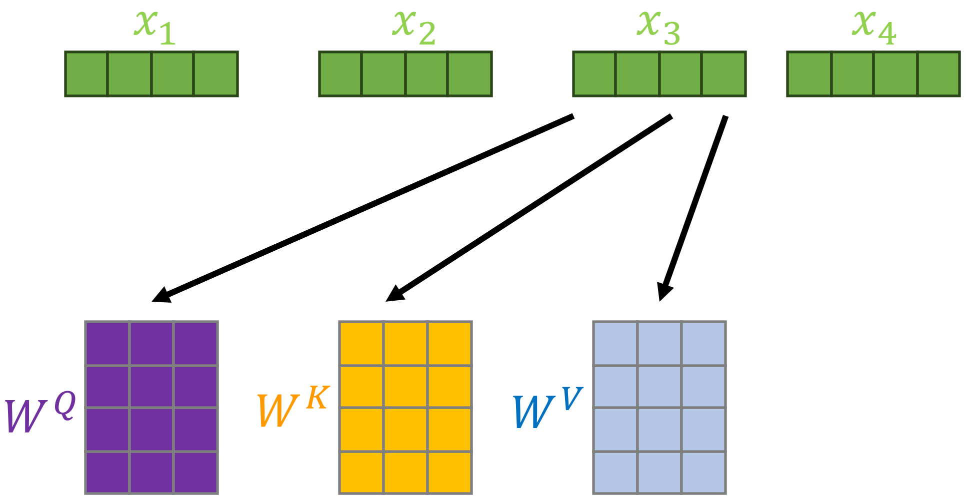


Query
$[0.3, 1.2, 2.2]^T$
$[0.1, 6.2, 2.1]^T$

Keys	Values
$[0.4, 1.4, 0]^T$	$[2.3, 1.1, 3.5]^T$
$[0.2, -1.1, 0]^T$	$[3.0, 1.0, 0.5]^T$

Idea:

- Every token makes:
  - Key
  - Value
  - Query

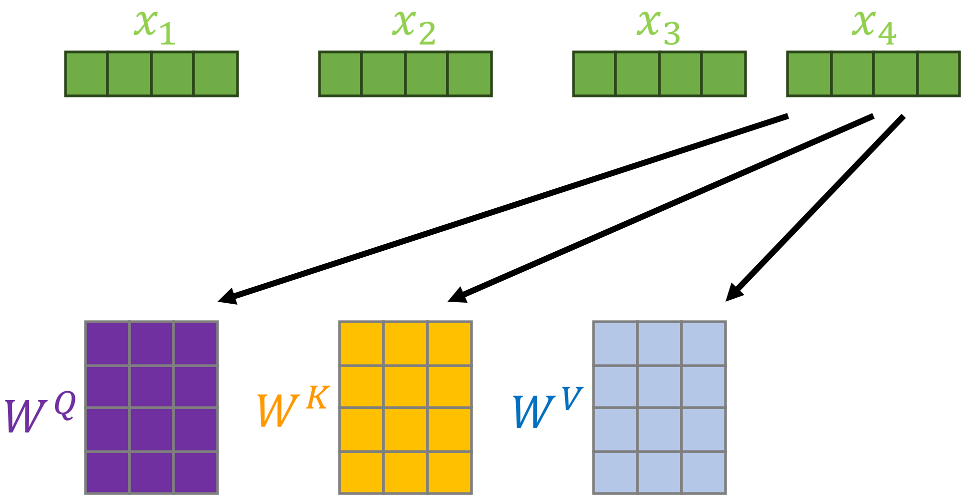


Idea:

- Every token makes:
  - Key
  - Value
  - Query

Query
$[0.3, 1.2, 2.2]^T$
$[0.1, 6.2, 2.1]^T$
$[2.1, 2.5, 5.2]^T$

Keys	Values
$[0.4, 1.4, 0]^T$	$[2.3, 1.1, 3.5]^T$
$[0.2, -1.1, 0]^T$	$[3.0, 1.0, 0.5]^T$
$[0.3, 1.2, 4]^T$	$[-2.2, 4.1, 2.5]^T$



Idea:

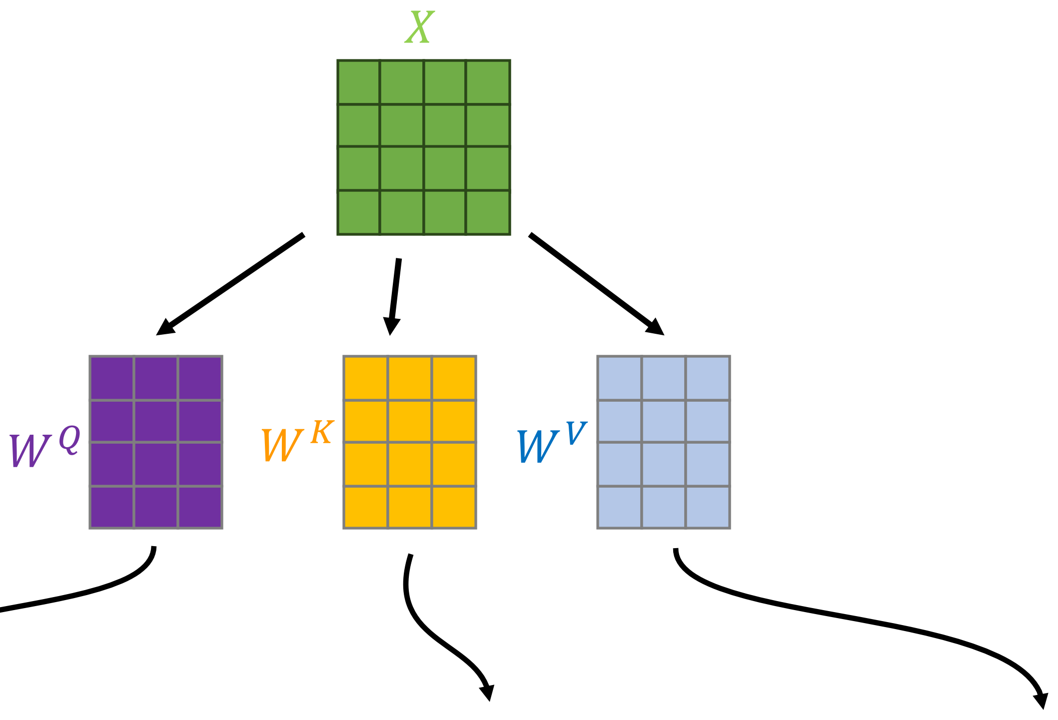
- Every token makes:
  - Key
  - Value
  - Query

Query
$[0.3, 1.2, 2.2]^T$
$[0.1, 6.2, 2.1]^T$
$[2.1, 2.5, 5.2]^T$
$[1.1, 6.6, 4.3]^T$

Keys	Values
$[0.4, 1.4, 0]^T$	$[2.3, 1.1, 3.5]^T$
$[0.2, -1.1, 0]^T$	$[3.0, 1.0, 0.5]^T$
$[0.3, 1.2, 4]^T$	$[-2.2, 4.1, 2.5]^T$
$[0.3, 1.2, 4]^T$	$[-2.2, 4.1, 2.5]^T$

Idea:

- Every token makes:
  - Key
  - Value
  - Query



Query
$[0.3, 1.2, 2.2]^T$
$[0.1, 6.2, 2.1]^T$
$[2.1, 2.5, 5.2]^T$
$[1.1, 6.6, 4.3]^T$

Keys	Values
$[0.4, 1.4, 0]^T$	$[2.3, 1.1, 3.5]^T$
$[0.2, -1.1, 0]^T$	$[3.0, 1.0, 0.5]^T$
$[0.3, 1.2, 4]^T$	$[-2.2, 4.1, 2.5]^T$
$[0.3, 1.2, 4]^T$	$[-2.2, 4.1, 2.5]^T$

Query

Query
$[0.3, 1.2, 2.2]^T$

	Keys	Values
0	$[0.4, 1.4, 0]^T$	$[2.3, 1.1, 3.5]^T$
0	$[0.2, -1.1, 0]^T$	$[3.0, 1.0, 0.5]^T$
1	$[0.3, 1.2, 2.2]^T$	$[4.4, 1.4, 3.5]^T$
0	$[0.3, 1.2, 4]^T$	$[-2.2, 4.1, 2.5]^T$

Result
$[4.4, 1.4, 3.5]^T$

$$z = \sum_{j=1}^{n} \mathbb{1}(\boldsymbol{q} = \boldsymbol{k}_j)\boldsymbol{v}_j$$

Relaxed Query

Query		Keys	Values	Result
$[0.3, 1.2, 2.2]^T$	.1	$[0.4, 1.4, 0]^T$	$[2.3, 1.1, 3.5]^T$	$[4.4, 1.4, 3.5\ ]^T$
	.1	$[0.2, -1.1, 0]^T$	$[3.0, 1.0, 0.5]^T$	
	.7	$[0.3, 1.2, 2.2]^T$	$[4.4, 1.4, 3.5]^T$	
	.1	$[0.3, 1.2, 4]^T$	$[-2.2, 4.1, 2.5]^T$	

$$\boldsymbol{z} = \sum_{j=1}^{n} \text{score}_j \boldsymbol{v}_j$$

$$\text{score}_j = \text{softmax}\Big(\underbrace{\text{similarity}(q, \boldsymbol{k}_j)}_{\frac{\boldsymbol{q}^T \boldsymbol{k}_j}{\sqrt{\boldsymbol{d}_k}}}\Big)$$

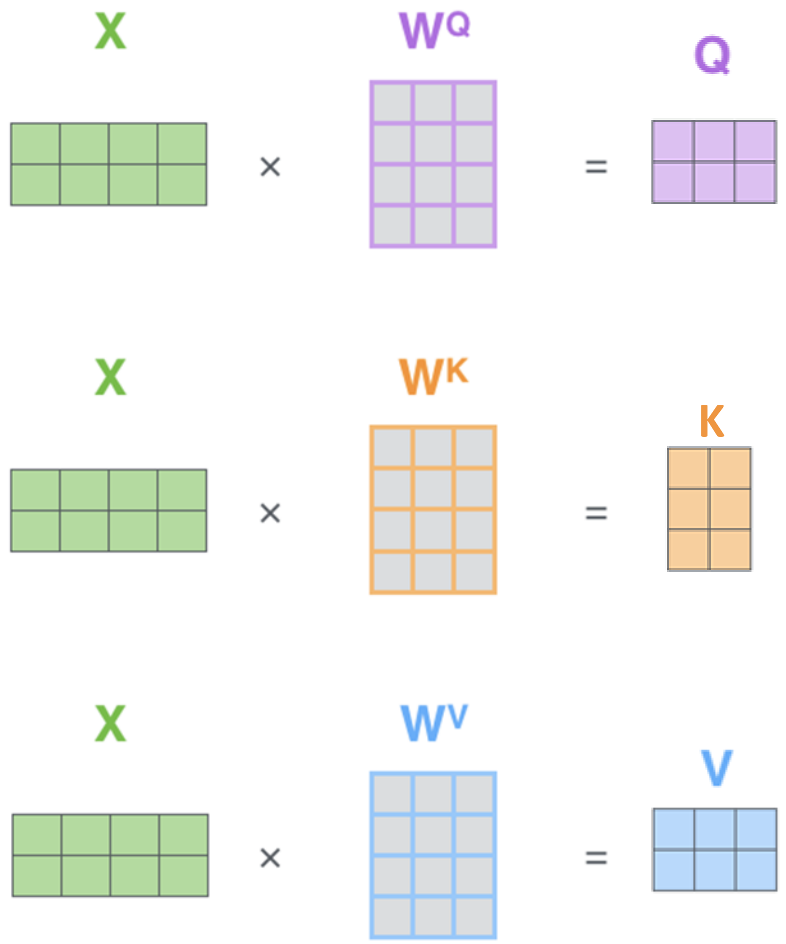
https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Self-Attention

Application to sequence: matrix multiplication

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

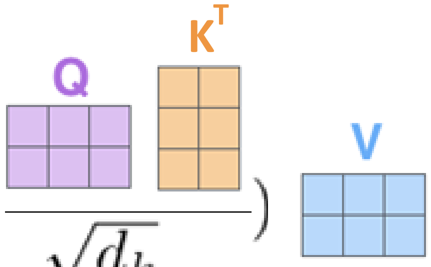
Q, **K**, **V** are computed from **X** (embedding of input sequence) with *learned* weight matrix



https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

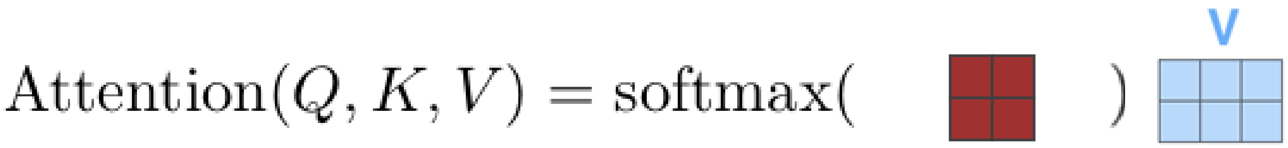
Self-Attention

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



https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Self-Attention

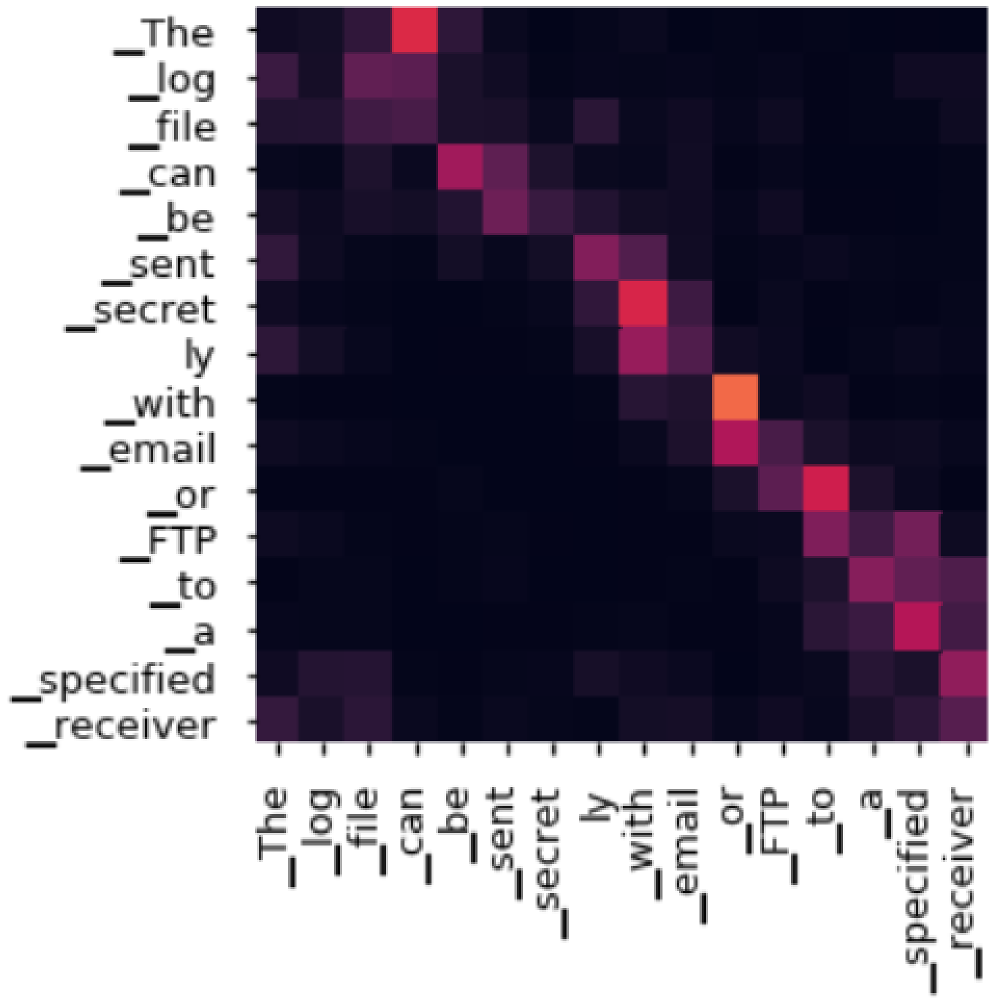


https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Self-Attention

$$\text{Attention}(Q, K, V) =$$

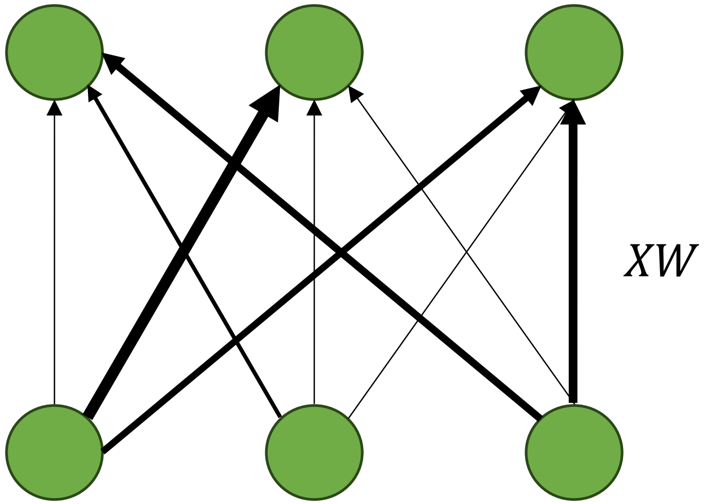
Attention Matrix



Self-Attention vs. Feed-Forward

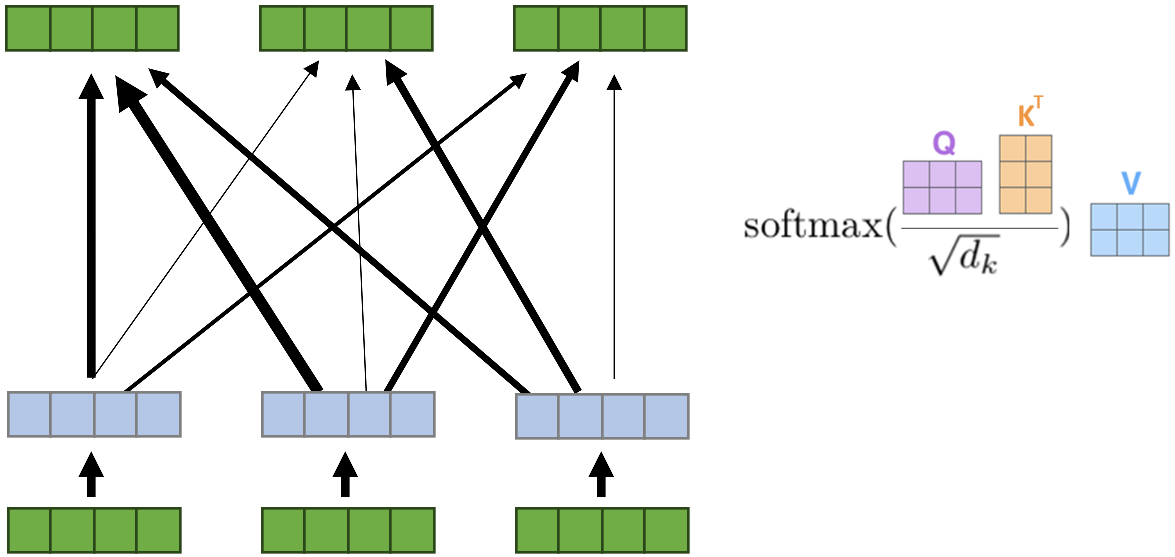
Perceptron / Feed-Forward:

- **Fixed weight matrix**



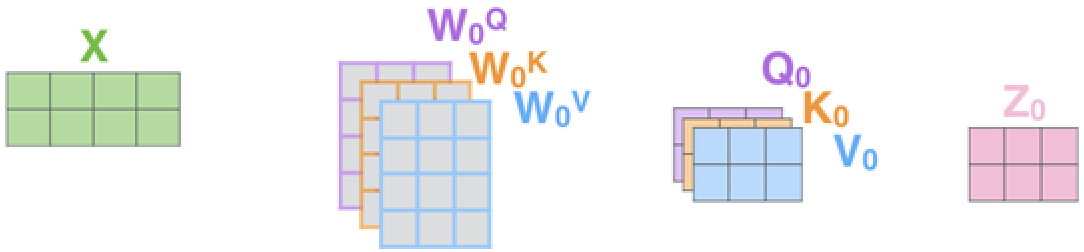
Self-Attention:

- **Dynamic weight matrix**
- Computed form inputs



https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

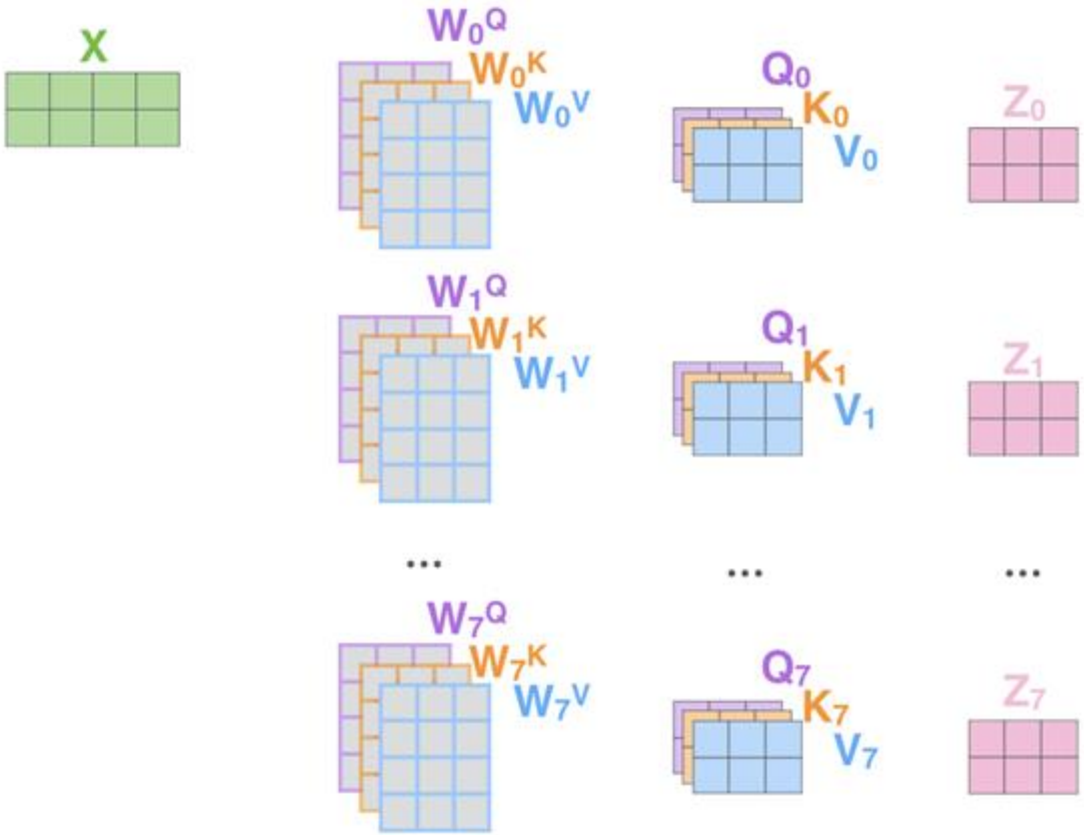
Multi-headed self-attention



https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Multi-headed self-attention

Multiple attention heads for increased model capacity

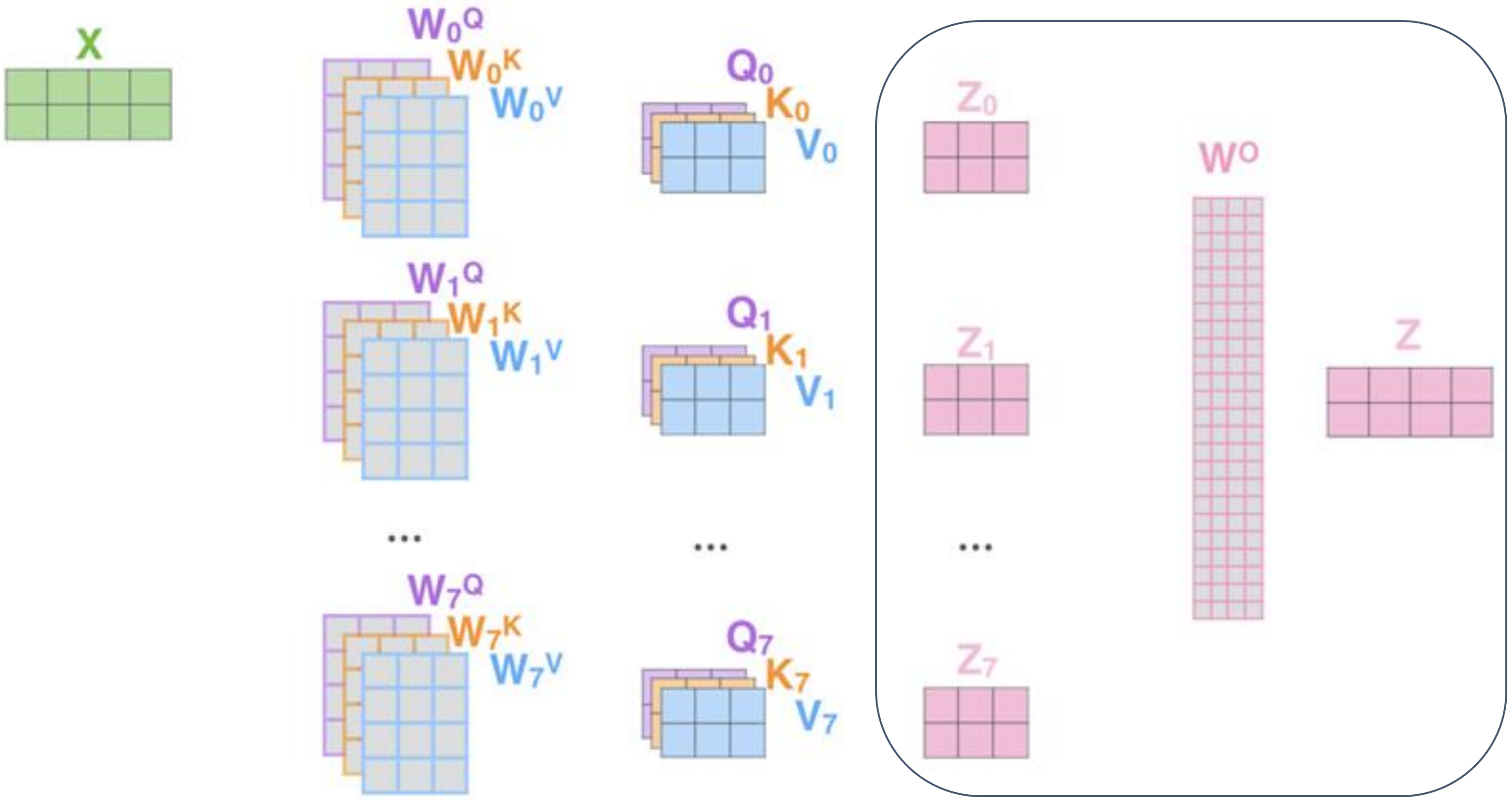


https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Multi-headed self-attention

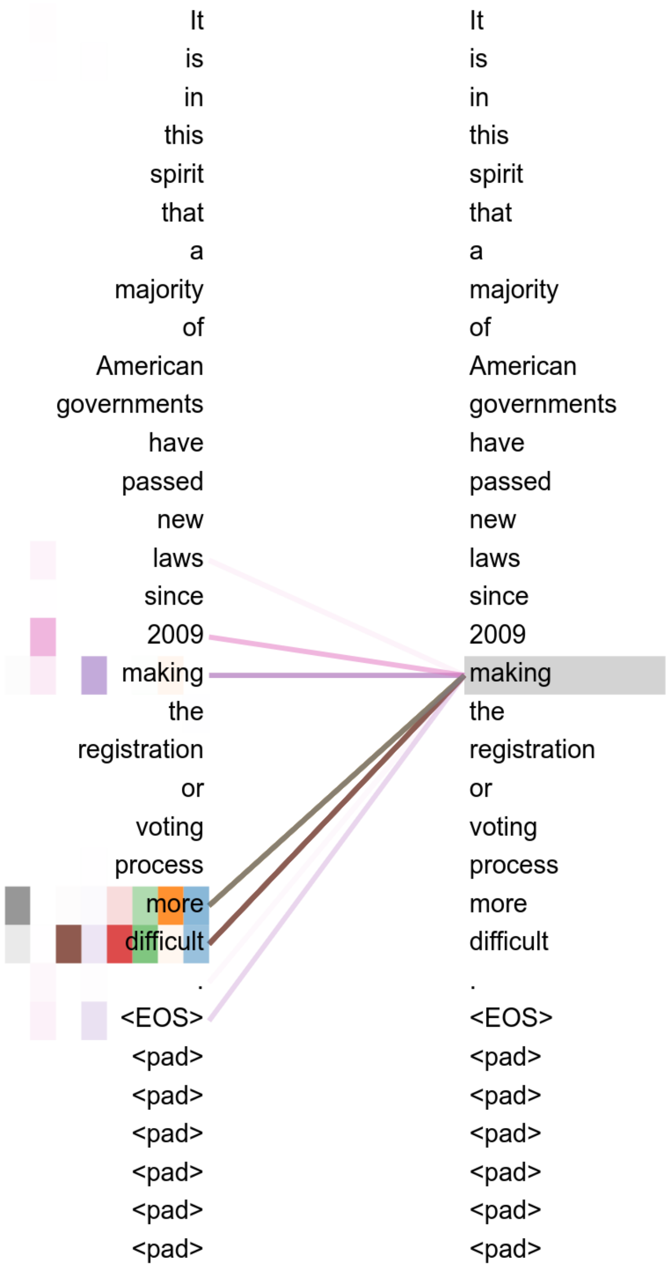
Multiple attention heads for increased model capacity

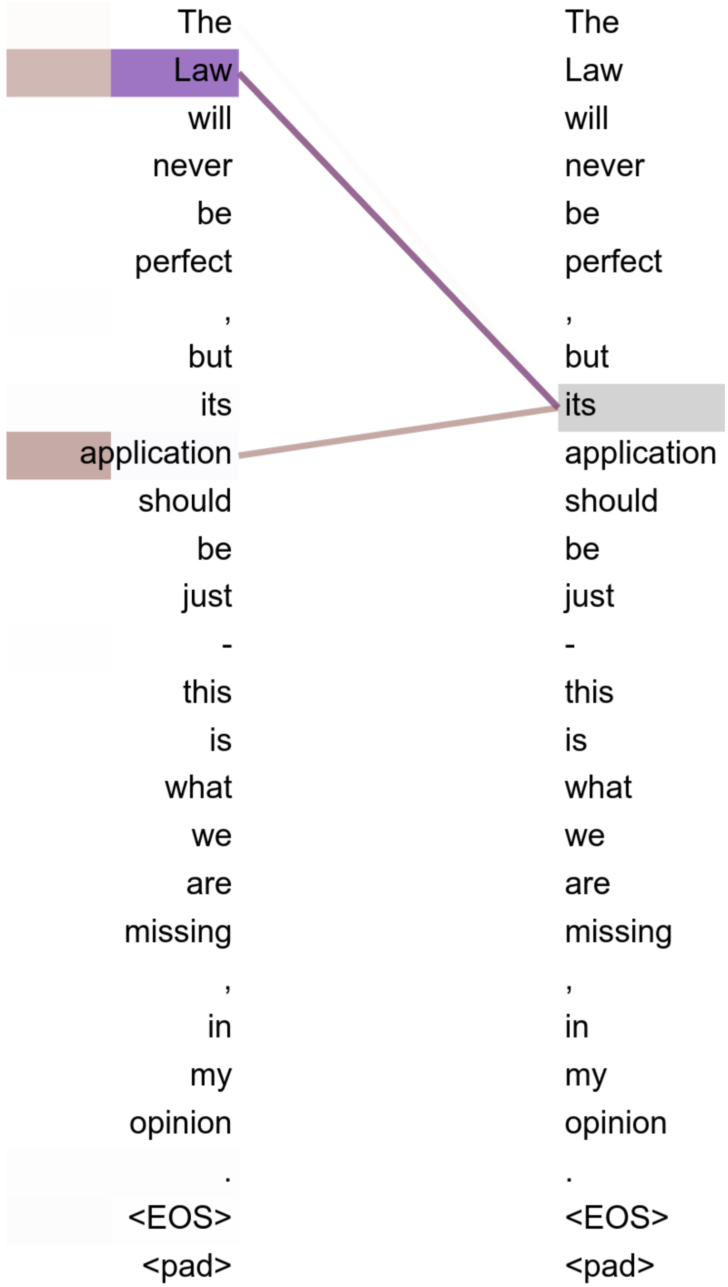
Combine with additional feed-forward layer



Concat z_i outputs, project to z with learned weight matrix

Ashish Vaswani et al. https://arxiv.org/pdf/1706.03762.pdf





Ashish Vaswani et al. https://arxiv.org/pdf/1706.03762.pdf

Next Video: the Decoder