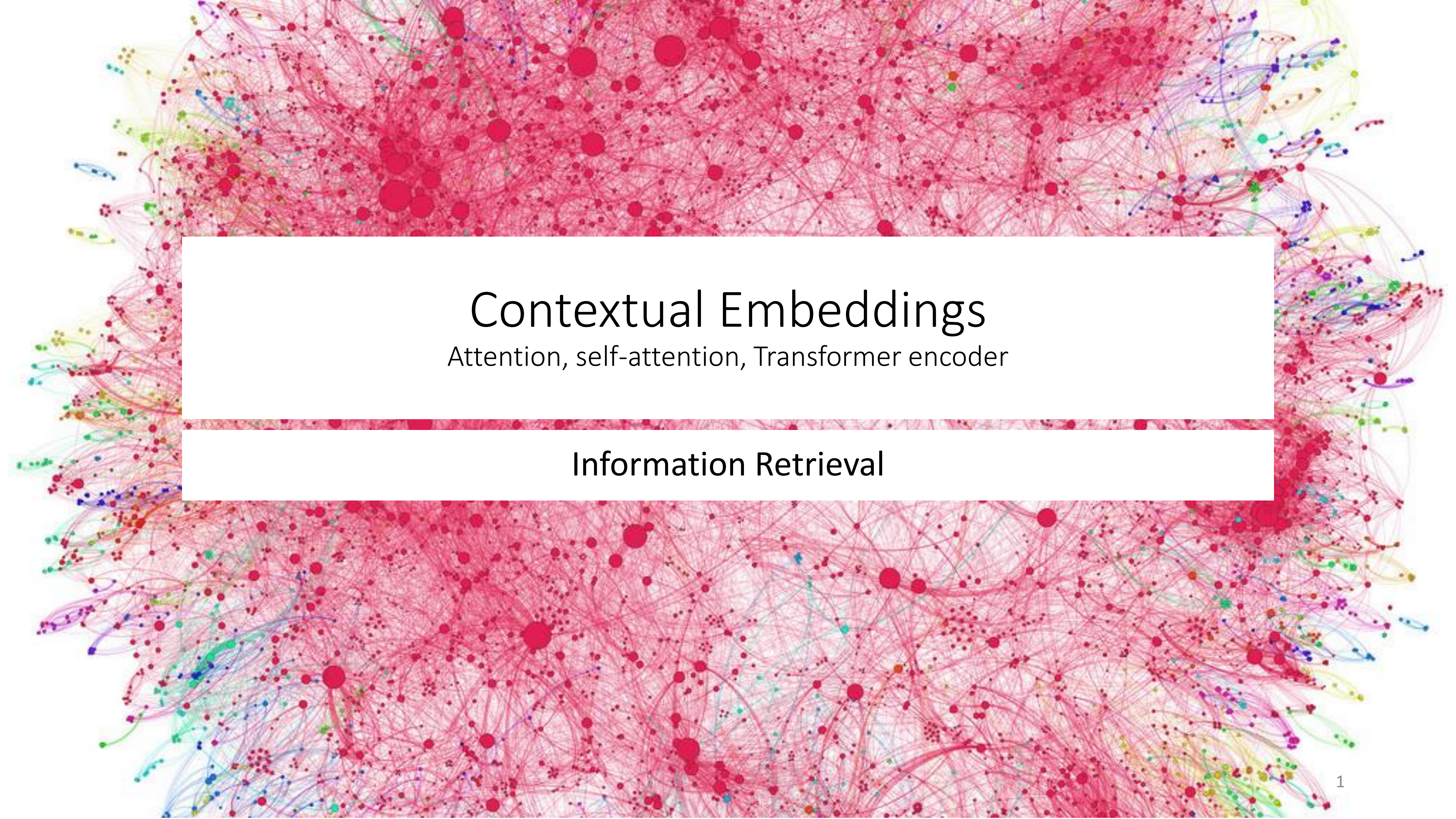
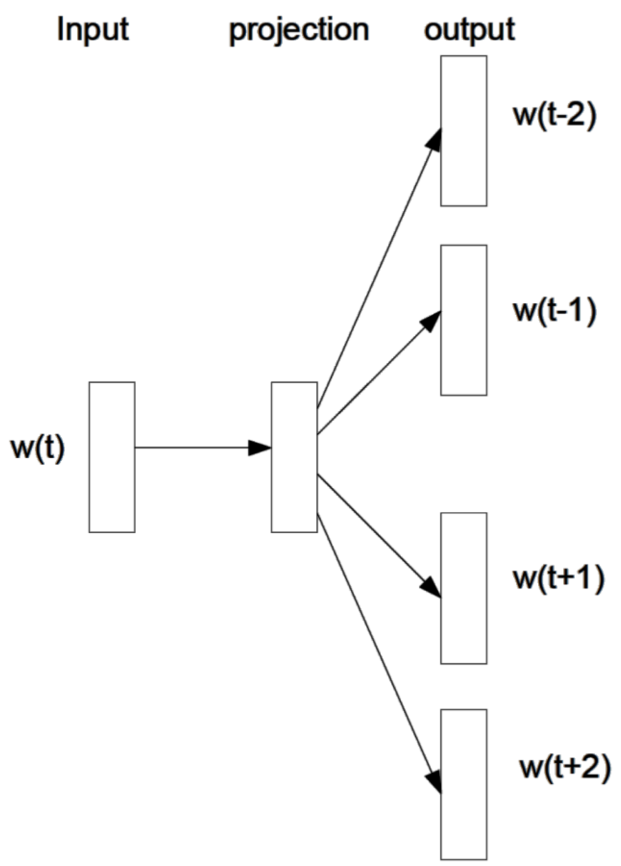


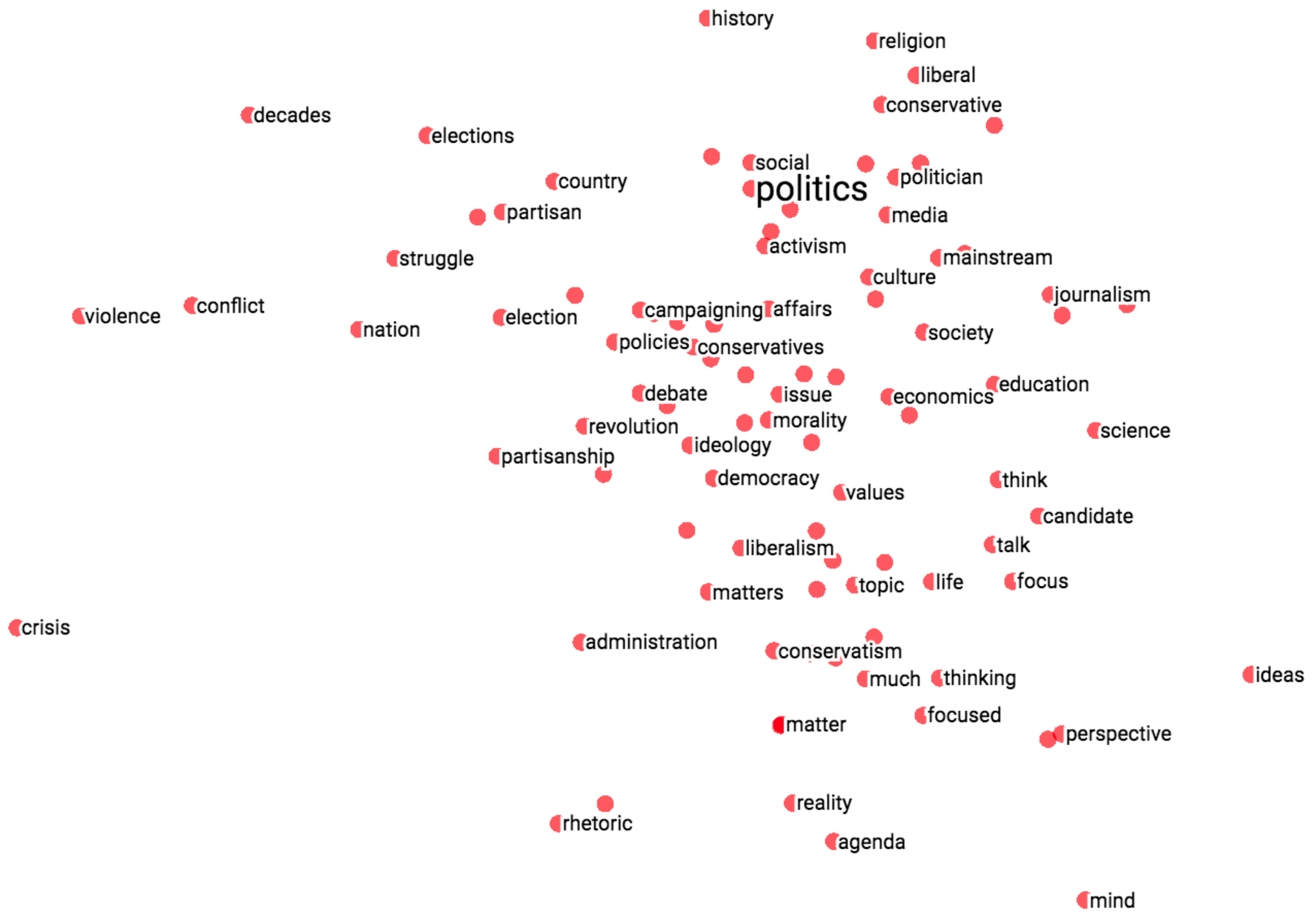
Contextual Embeddings

Attention, self-attention, Transformer encoder

Information Retrieval

Word embeddings (recap)





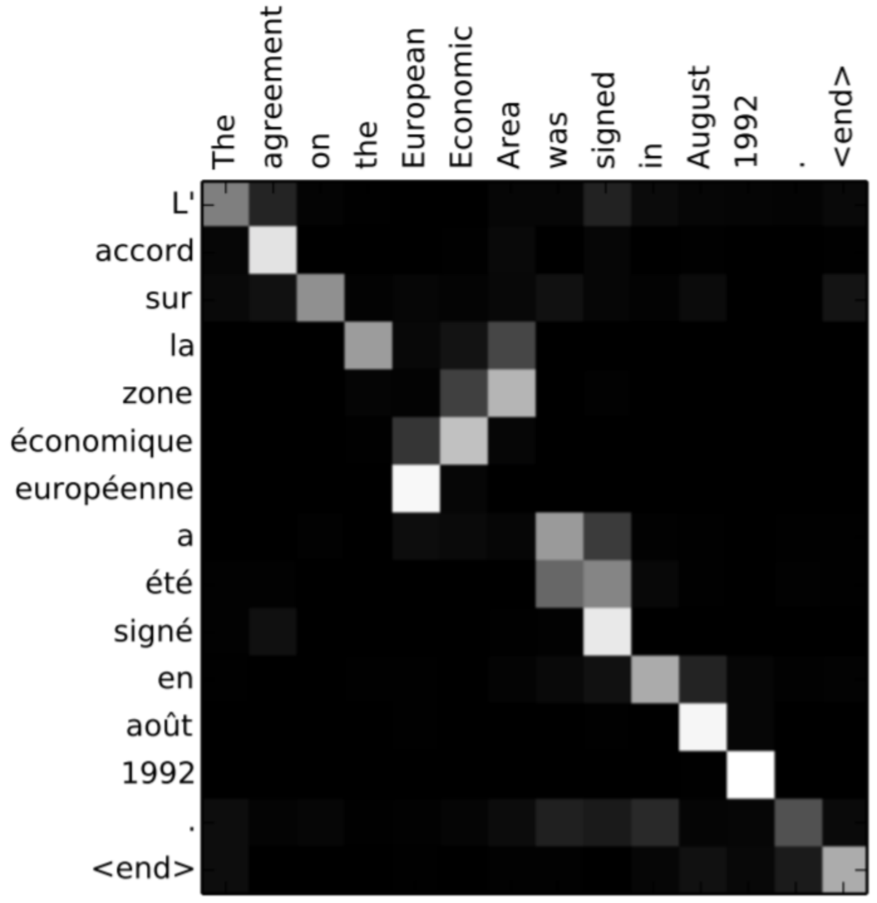
Limitations

- Word tokens
- Neighboring word prediction
- Static embeddings

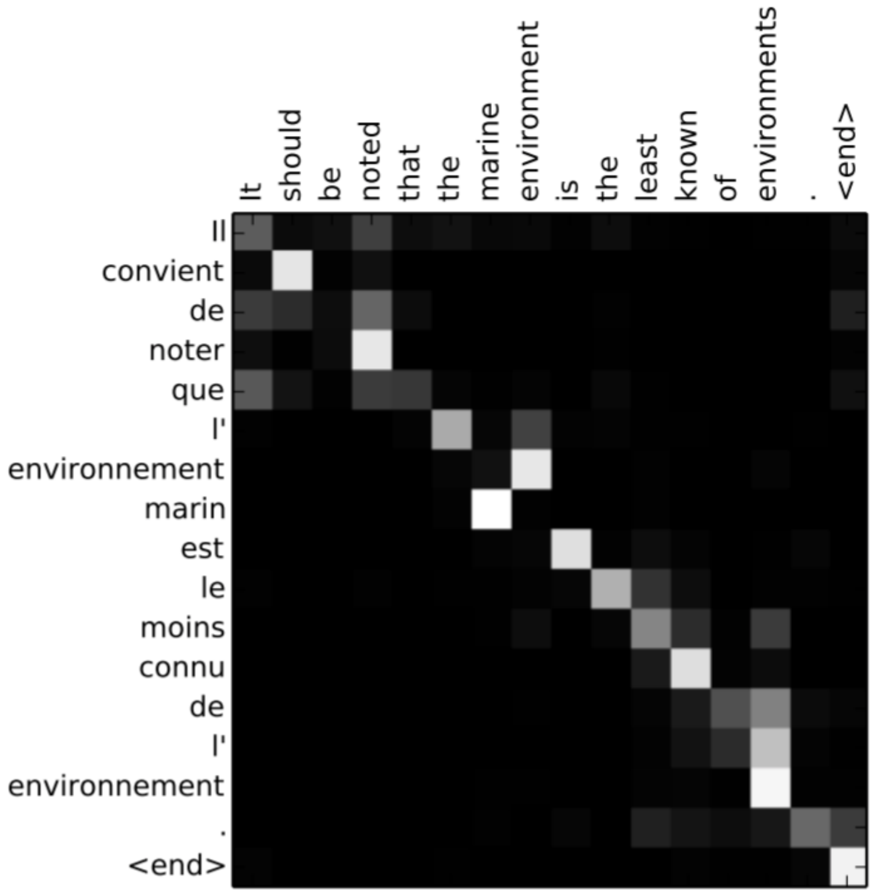
New words and different word spellings

- Limited to only seen words.
- Sub-word patterns (prefixes and suffixes) are ignored.
- Words with different spellings.

Relation between word embeddings



(a)

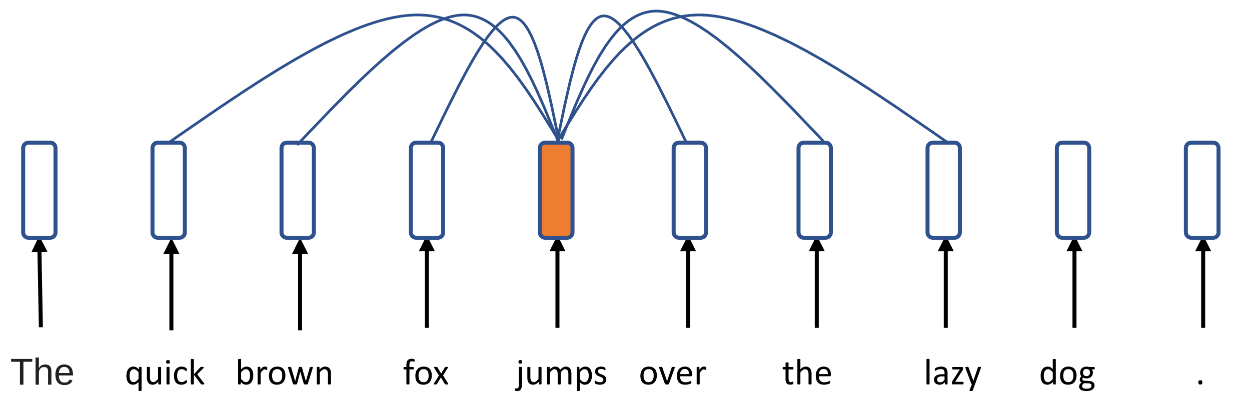


(b)

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*. https://arxiv.org/pdf/1409.0473.pdf

Words are used in different contexts

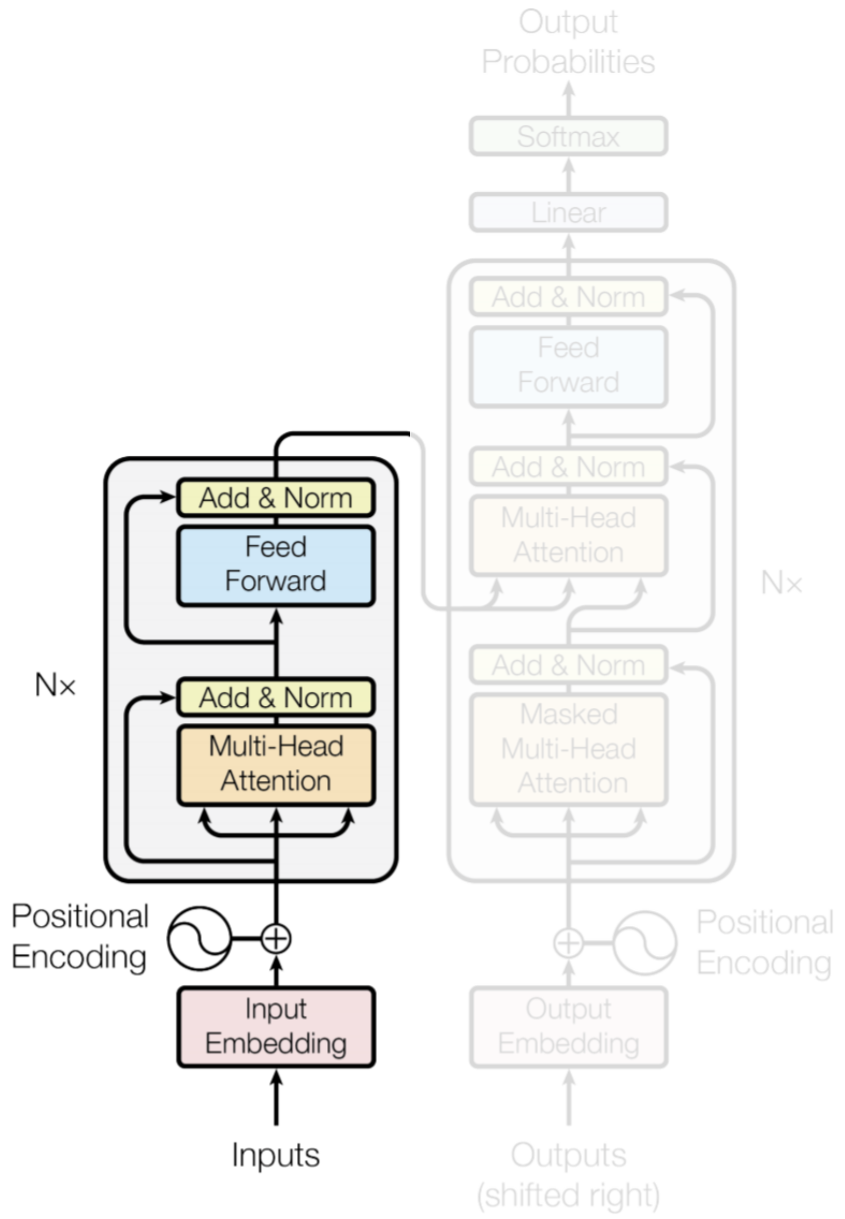
- Static word embeddings do not capture the context of the sentence.
- The same word used in different contexts may have different purposes or meaning.



Each word embedding must attend to its neighboring words.

The Transformer

- The Transformer model explores the concept of self-attention mechanism.
- Self-attention over the entire sequence and in both directions.
- The feedforward layer models non-linearities.
- The stack of such elements allows capturing different granularities of data, i.e. similar to deep CNN architectures.



Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017, December). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 6000-6010). https://arxiv.org/abs/1706.03762

Subword tokenization

- Three common algorithms:
  - Byte-Pair Encoding (BPE) (Sennrich et al., 2016)
  - Unigram language modeling tokenization (Kudo, 2018)
  - WordPiece (Schuster and Nakajima, 2012)

- All have 2 parts:
  - A **token learner** that takes a raw training corpus and induces a vocabulary (a set of tokens).
  - A **token segmenter** that takes a raw test sentence and tokenizes it according to that vocabulary

Byte Pair Encoding (BPE) token learner

- Let vocabulary be the set of all individual characters

  = {A, B, C, D,…, a, b, c, d….}

- Repeat:
  - Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')
  - Add a new merged symbol 'AB' to the vocabulary
  - Replace every adjacent 'A' 'B' in the corpus with 'AB’.

- Until k merges have been done.

Byte Pair Encoding (BPE) Addendum

- Most subword algorithms are run inside space-separated tokens.
- So we commonly first add a special end-of-word symbol '__' before space in training corpus
- Next, separate into letters.

BPE token learner

- Original (very fascinating 🙄) corpus:
- low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new
- Add end-of-word tokens, resulting in this vocabulary:

corpus	
5	l o w _
2	l o w e s t _
6	n e w e r _
3	w i d e r _
2	n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w

BPE

The next merges are:

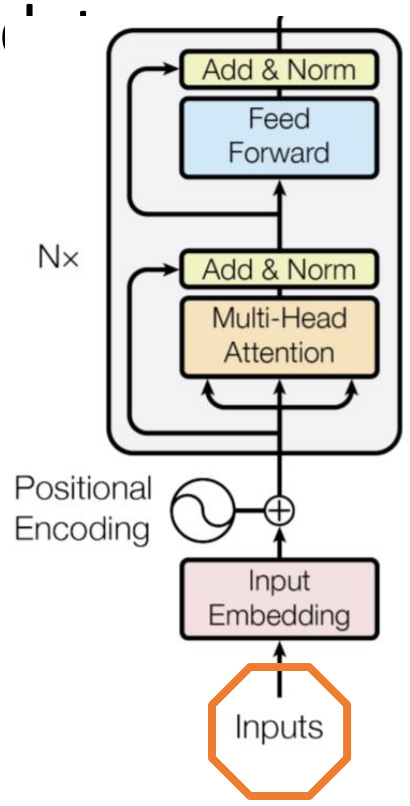
Merge	**Current Vocabulary**
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

BPE token segmenter algorithm

- On the test data, run each merge learned from the training
  - Greedily
  - In the order we learned them
  - (test frequencies don't play a role)

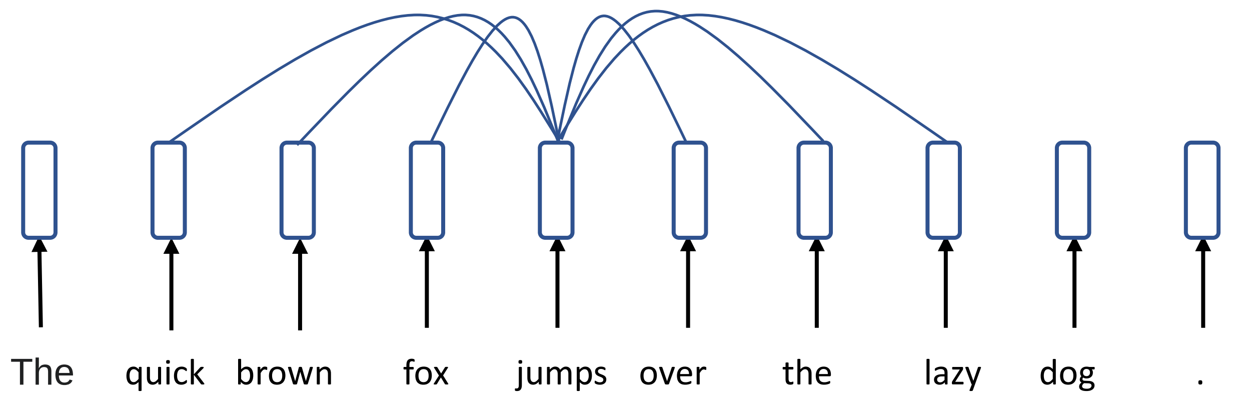
So: merge every e r to er, then merge er _ to er_, etc.

- Result:
  - Test set "n e w e r _" would be tokenized as a full word
  - Test set "l o w e r _" would be two tokens: "low er_"



Context embeddings

- Tokens show up on different positions of the sentence
- Token embeddings need to attend to the context

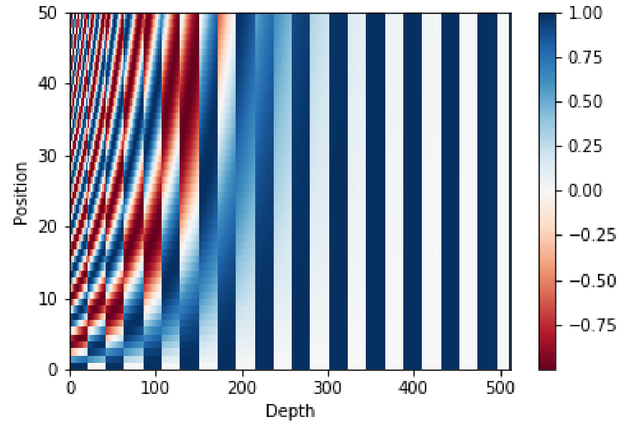


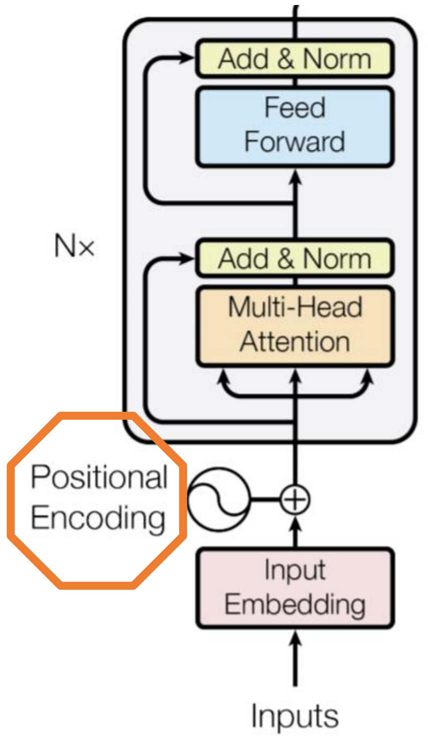
Contextual embeddings

Positional encodings + Self-attention

Positional encodings

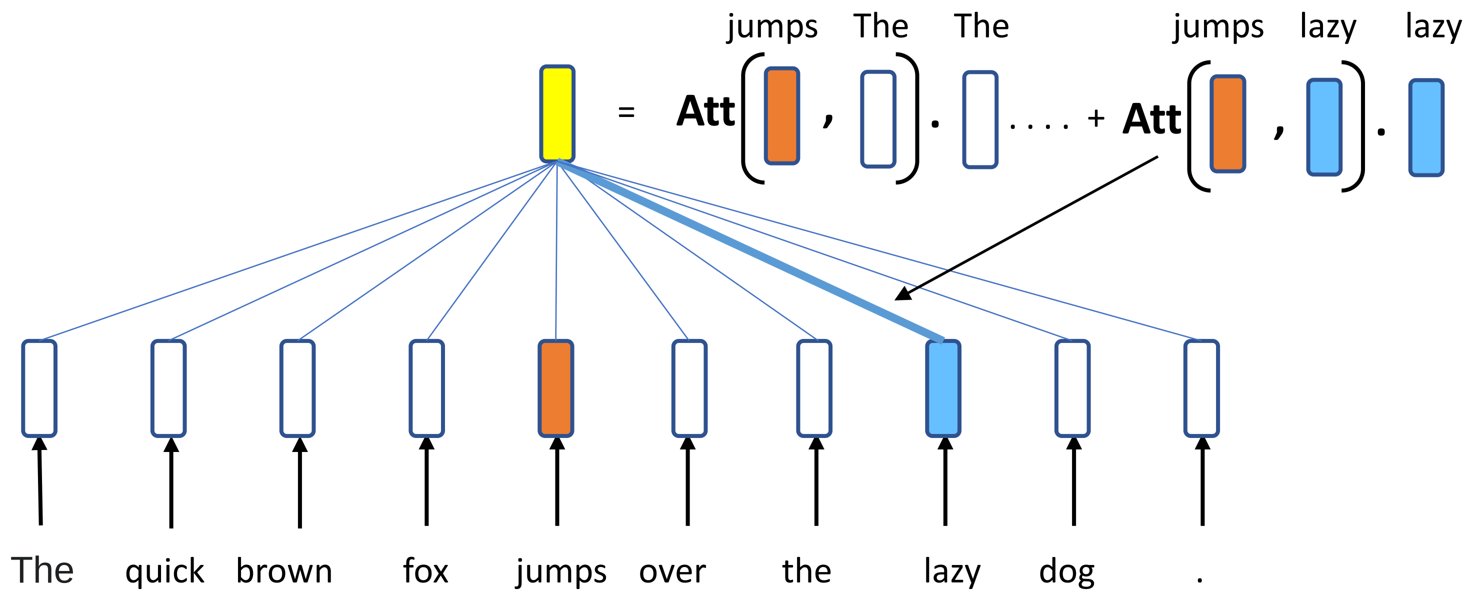
- The input embeddings follow the same idea of word embeddings.
- However, word embeddings do not capture sequence information.
- Positional embeddings, are associated to the position in the sequence.
- <u>Summing the word embedding to the positional will move the word embeddings according to the its position in the sentence.</u>





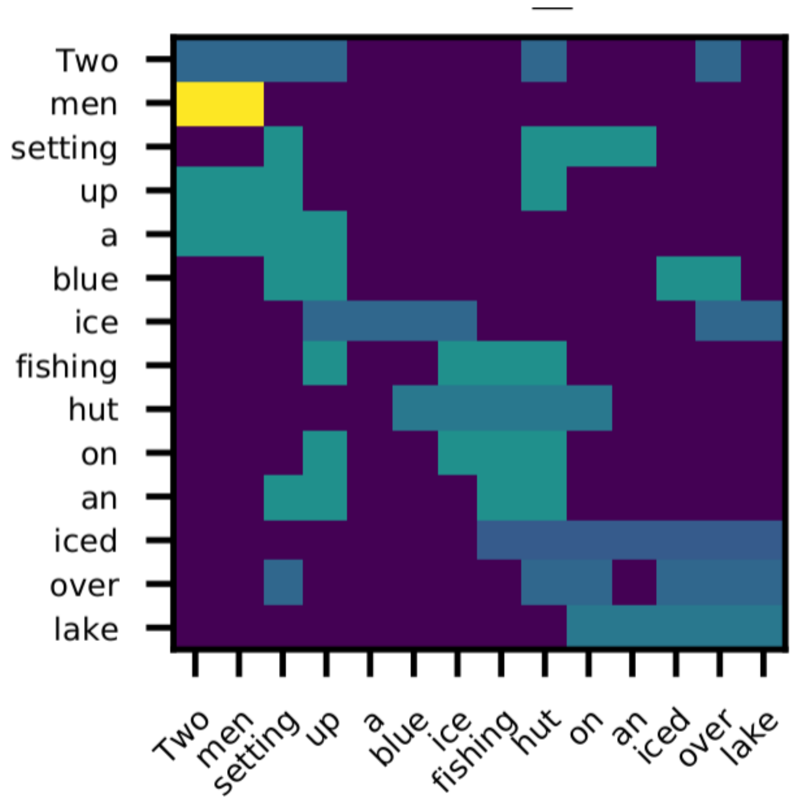
Attending to context

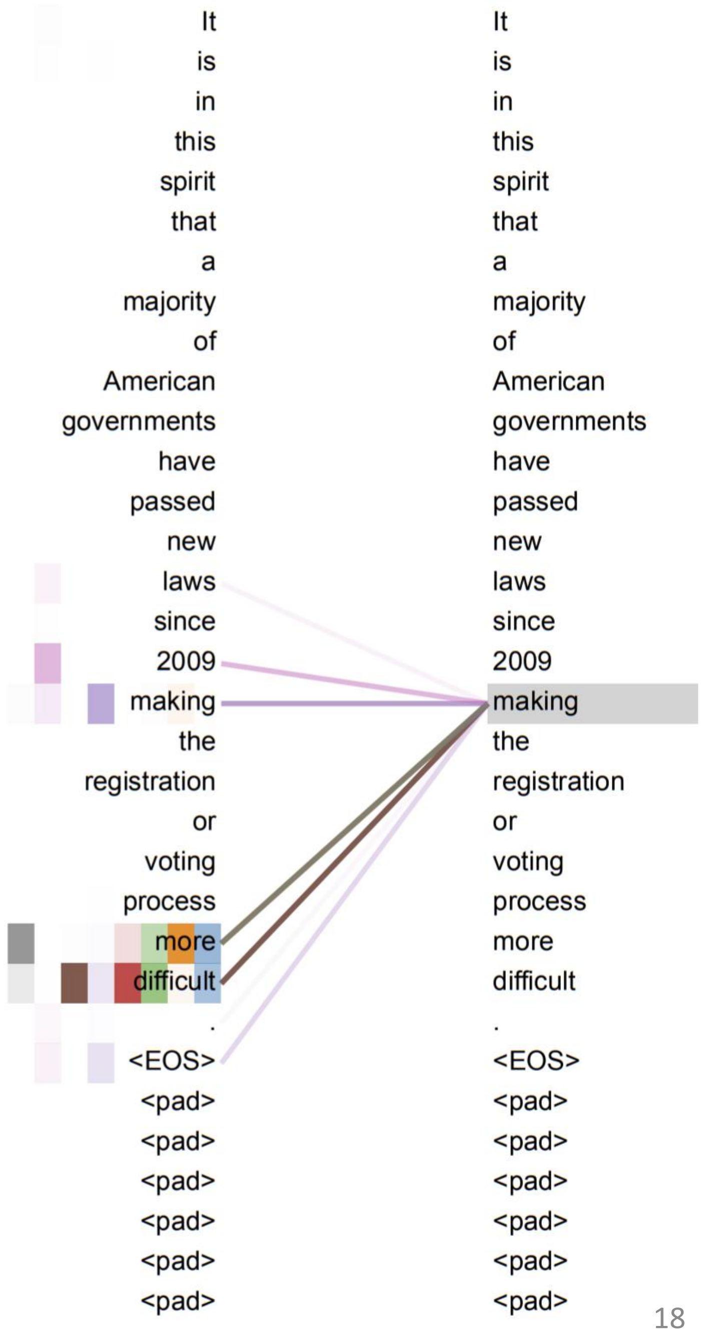
- Idea: compute token embeddings in way that it “attends” to neighboring tokens
- The output embedding is then a weighted combination of all the previous embeddings



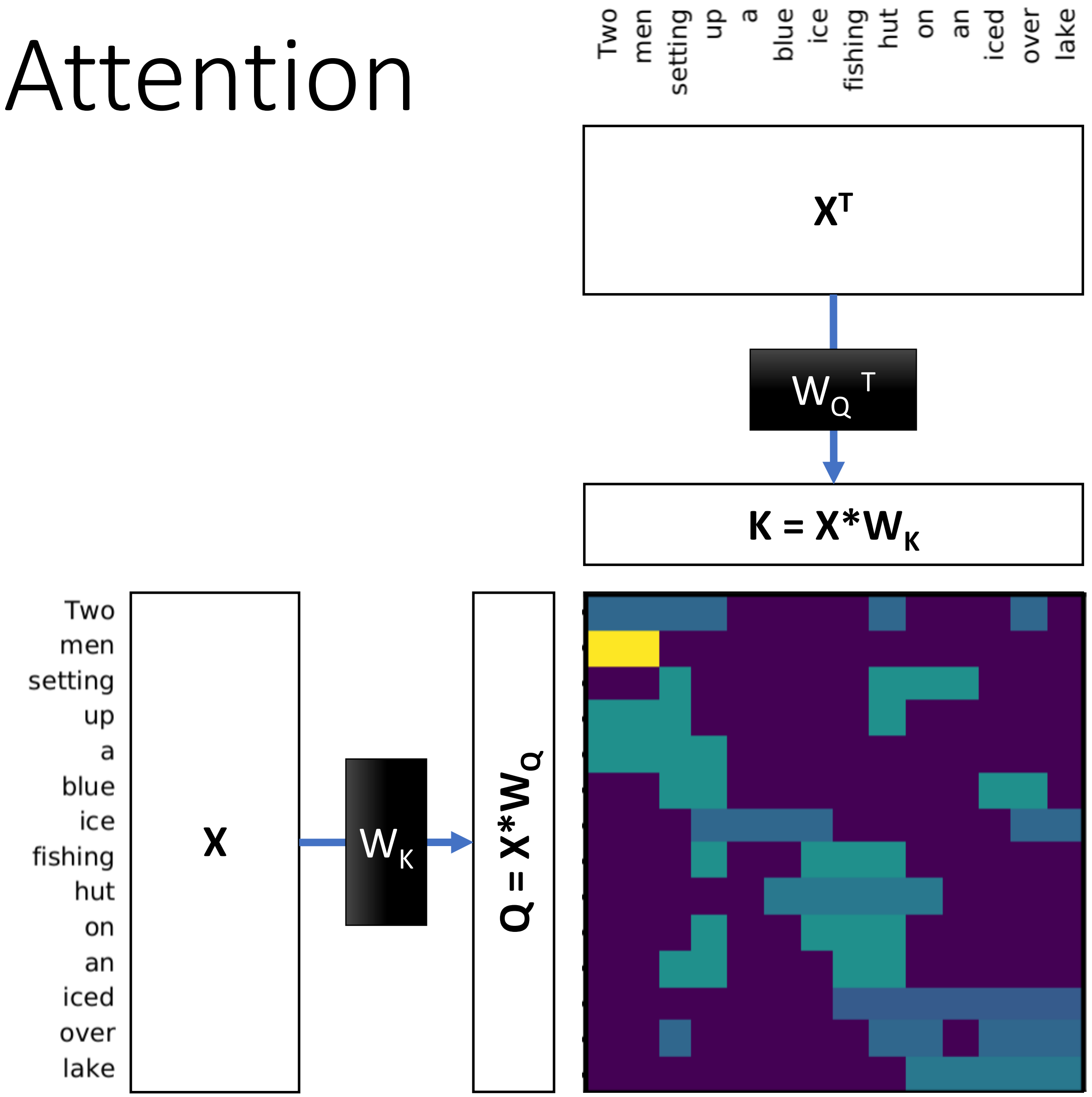
Self Attention

- Full self-attention is the central novelty of the Transformer





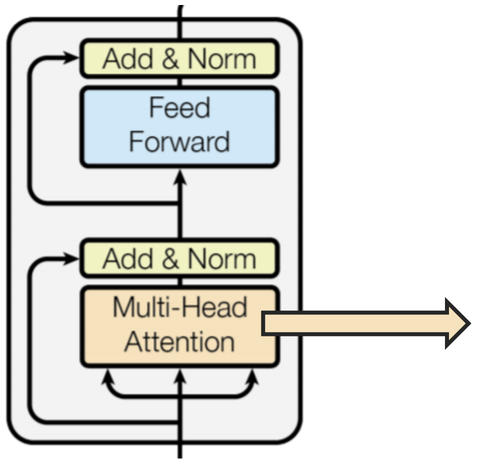
Self Attention



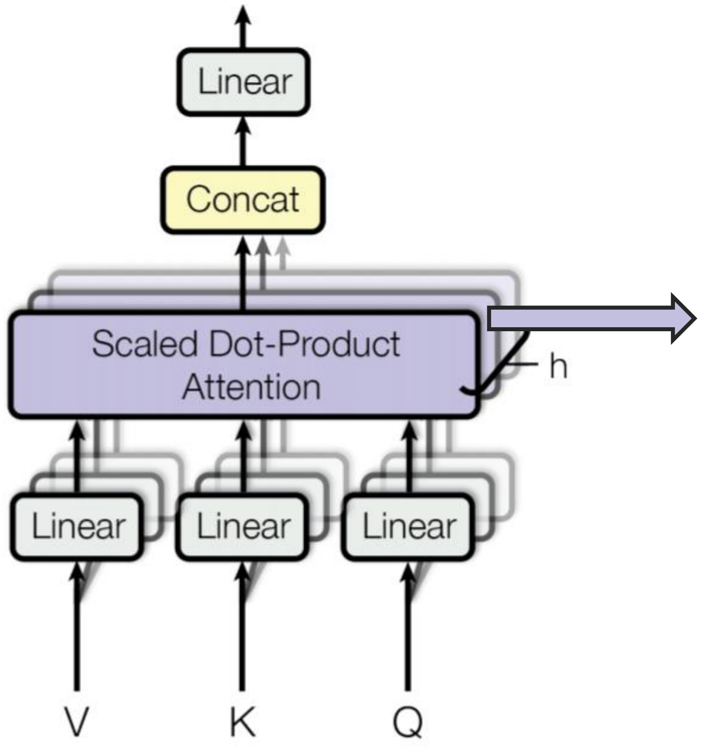
X = Input token embeddings

Transformer Layer Overview

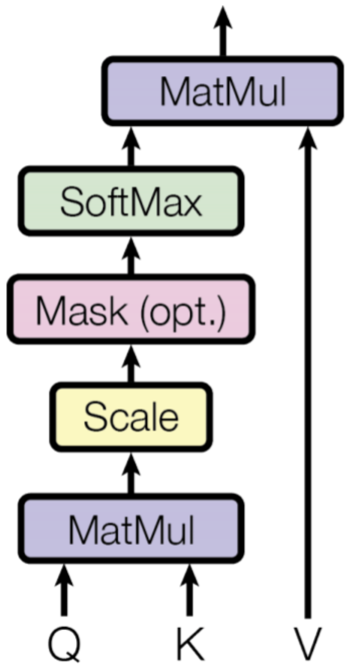
Transformer Layer
(there are N layers)



Multi-Head Attention
(there are h heads)

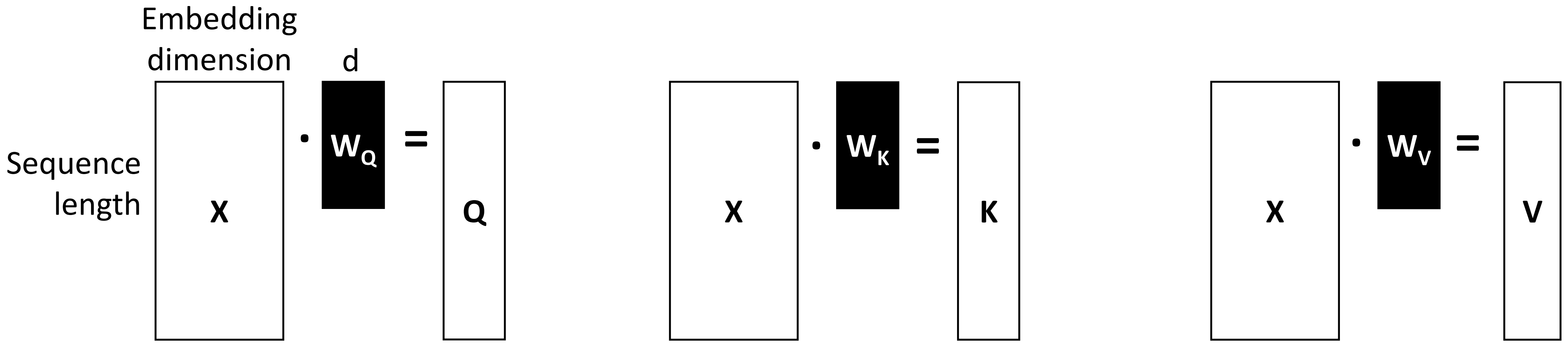


Single-Head Attention

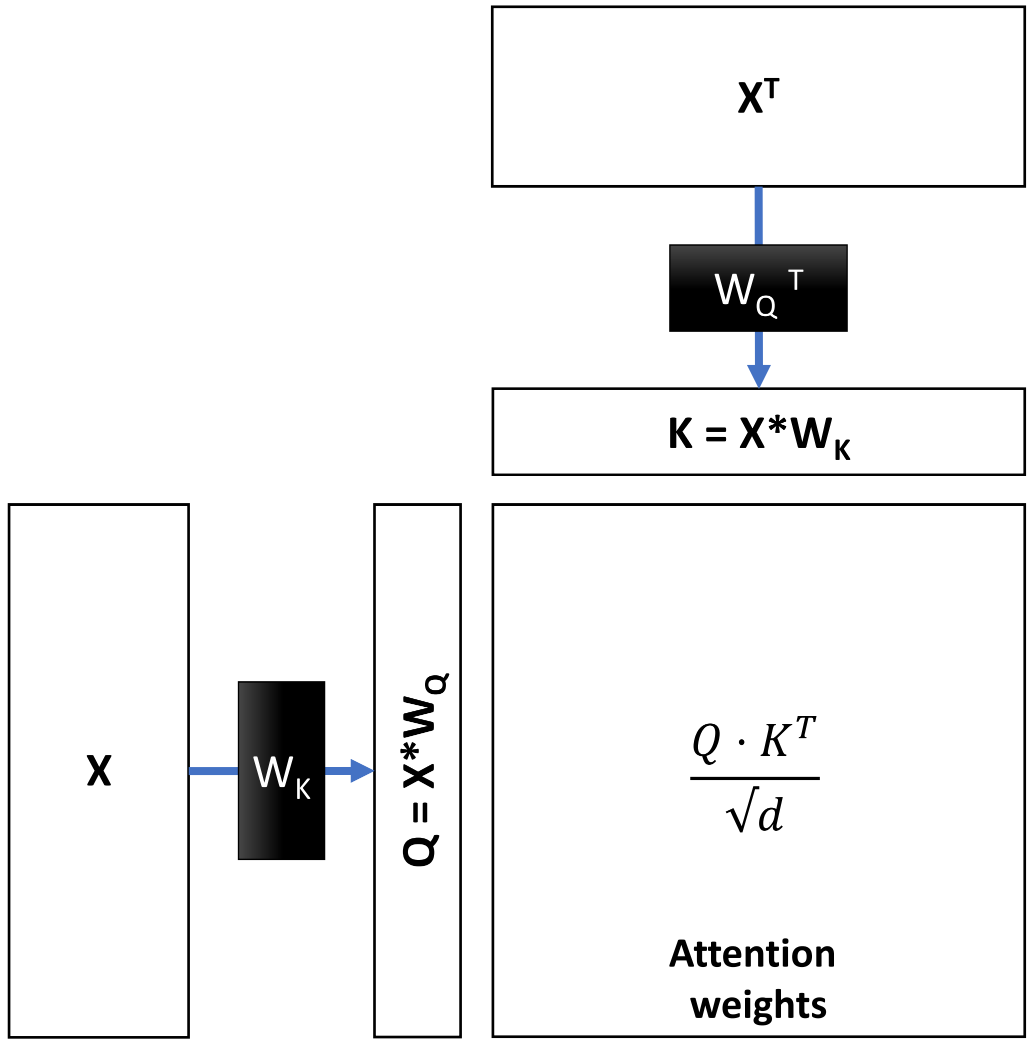


Projection matrices

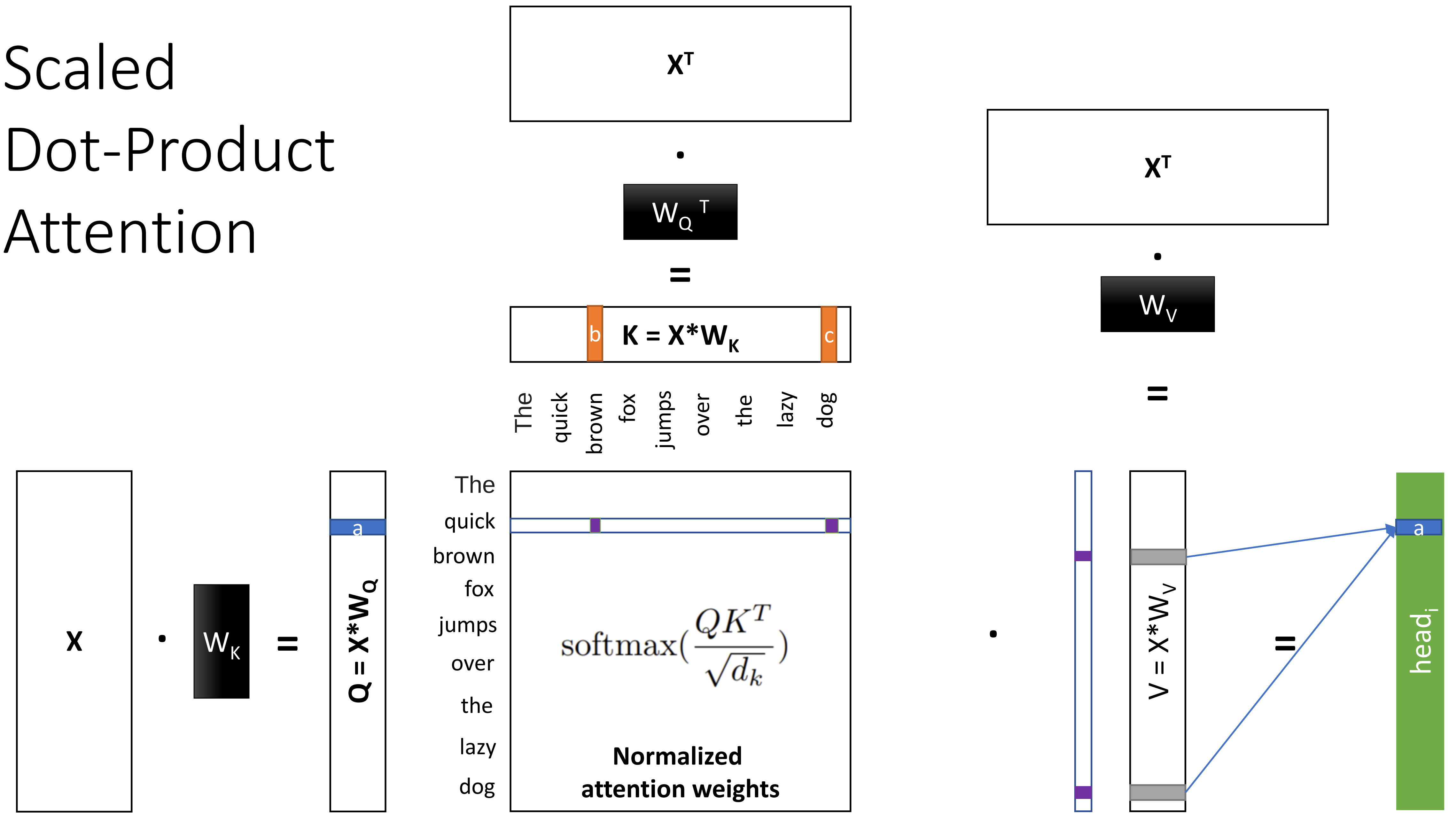
- X is a *sequence* of token embeddings corresponding to a sentence.
- These embeddings can be pre-trained, e.g. w2v or glove.



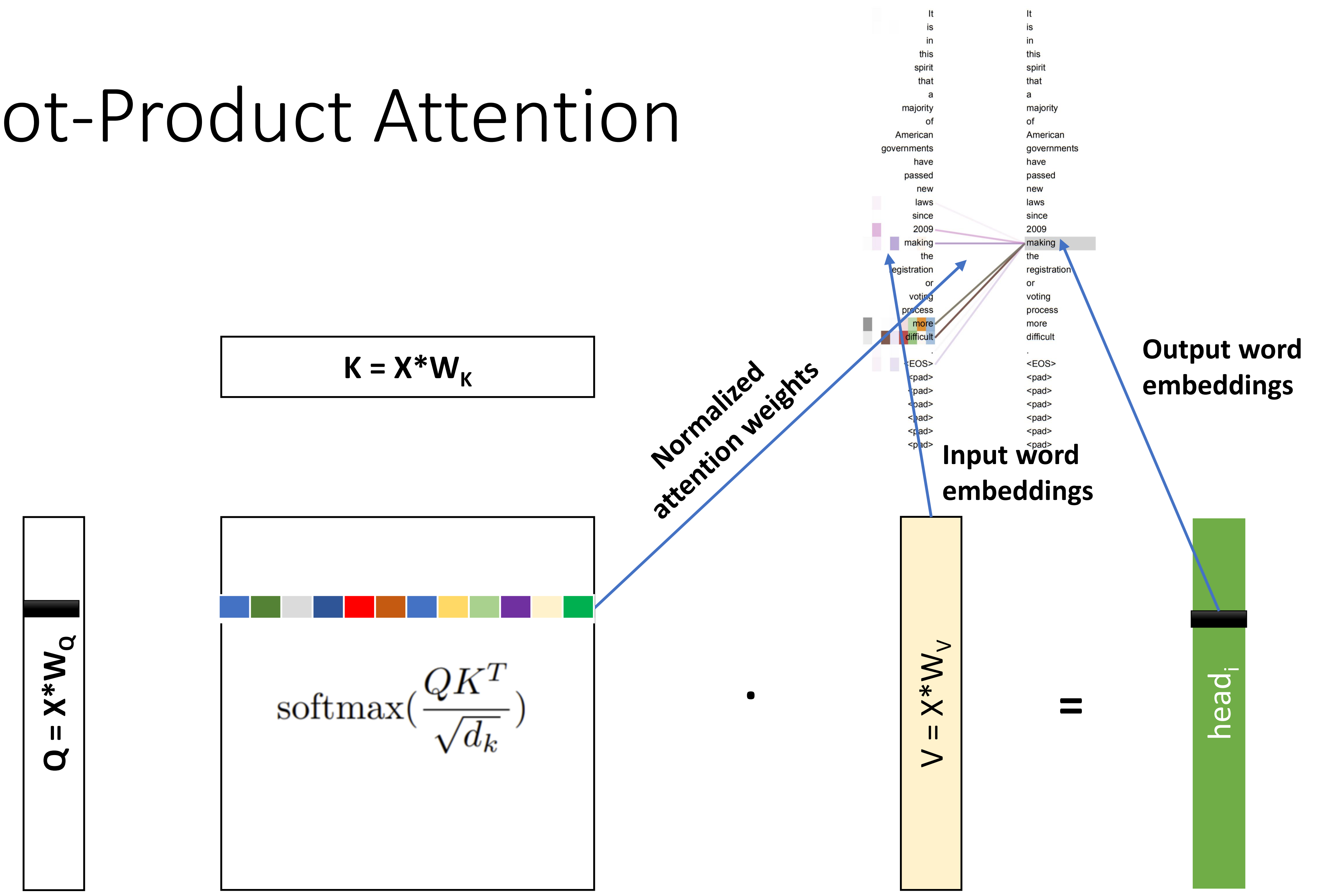
Self Attention



Scaled Dot-Product Attention



Scaled Dot-Product Attention

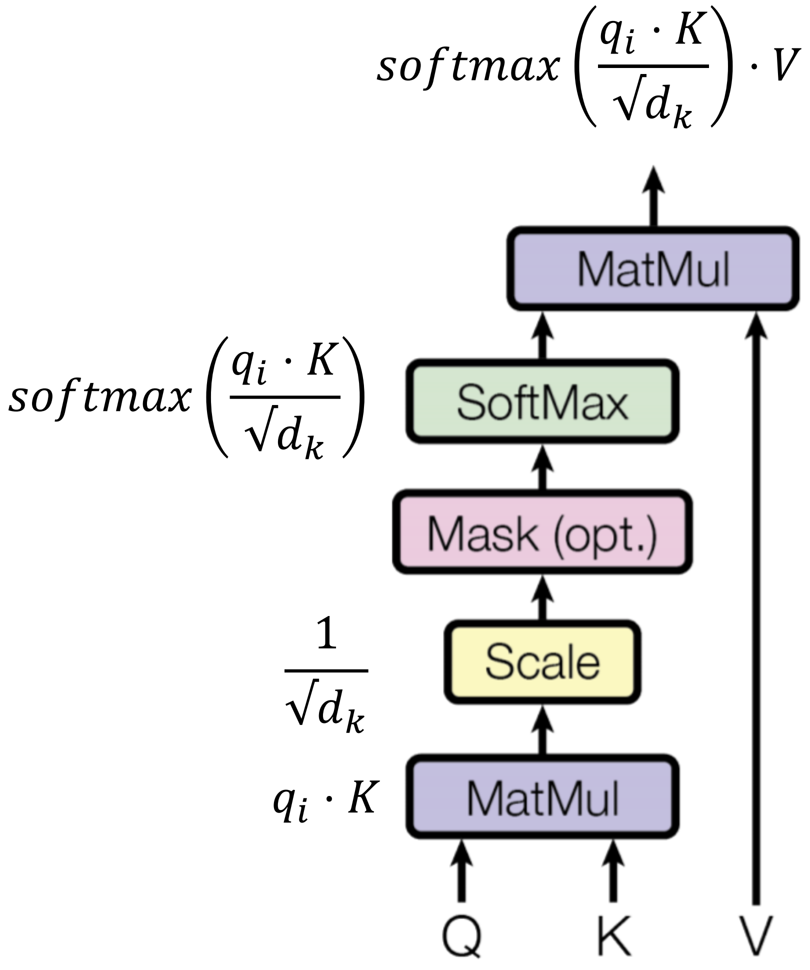


It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult . <EOS> <pad> <pad> <pad> <pad> <pad> <pad>
It is in this spirit that a majority of American governments have passed new laws since 2009 making the registration or voting process more difficult . <EOS> <pad> <pad> <pad> <pad> <pad> <pad>
K = X*W_K
Normalized attention weights
Output word embeddings
Input word embeddings
Q = X*W_Q
softmax(QK^T/√d_k)
·
V = X*W_V
=
head_i

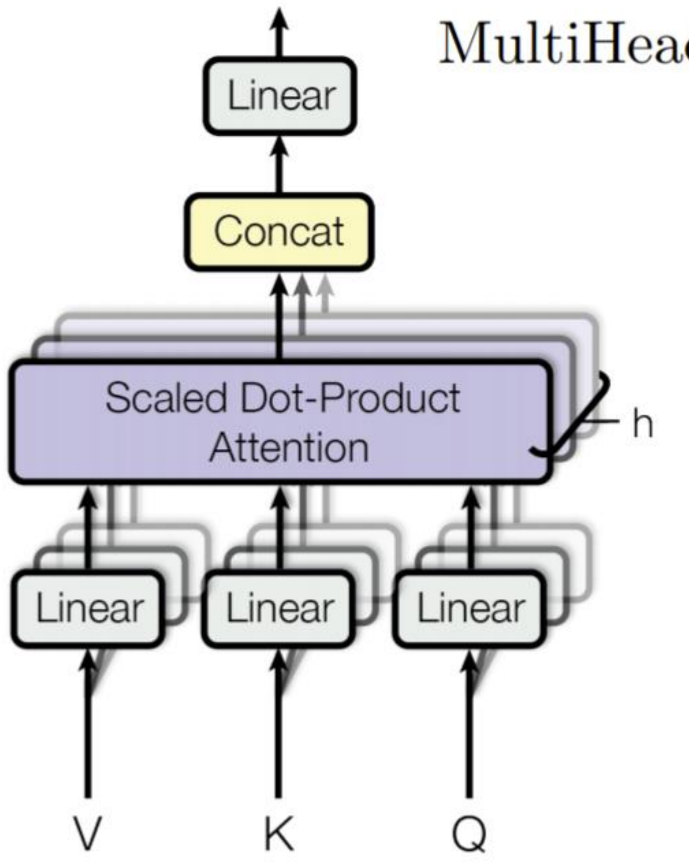
Scaled Dot-Product Attention

- The scaled dot-product attention lets each word attend to all other words
    - (let's ignore the mask for now)

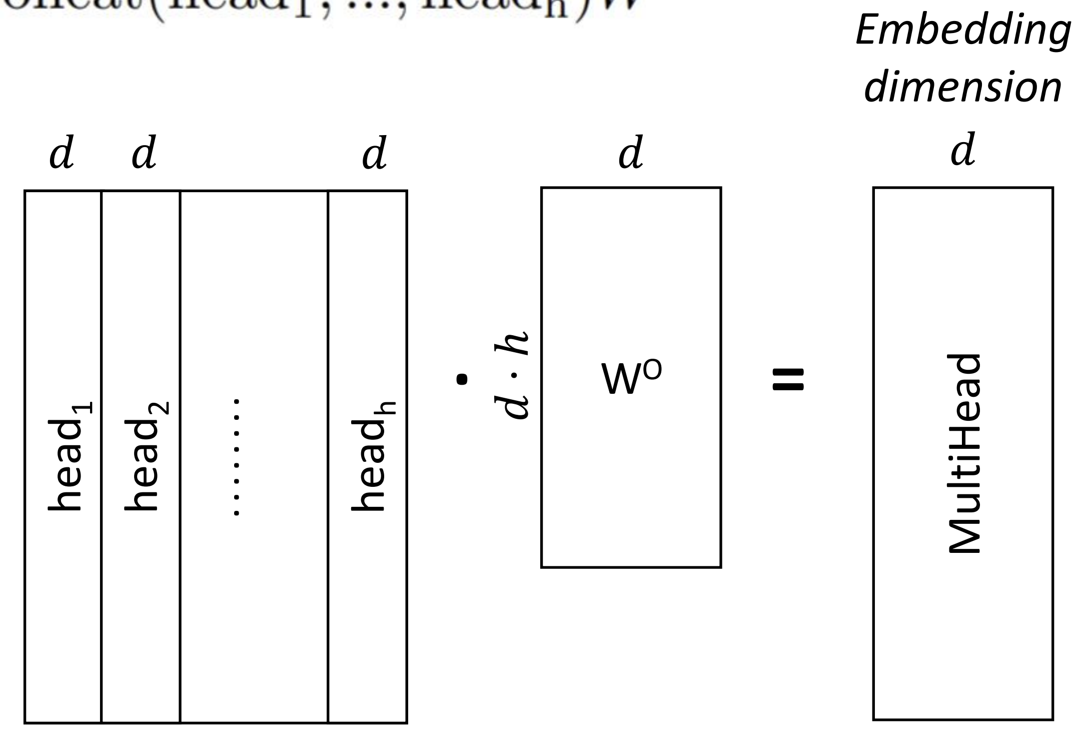
- $o_i = softmax\left(\frac{q_i \cdot K}{\sqrt{d_k}}\right) \cdot V$


$softmax\left(\frac{q_i \cdot K}{\sqrt{d_k}}\right) \cdot V$
MatMul
$softmax\left(\frac{q_i \cdot K}{\sqrt{d_k}}\right)$
SoftMax
Mask (opt.)
$\frac{1}{\sqrt{d_k}}$
Scale
$q_i \cdot K$
MatMul
Q
K
V

Multi-head attention



$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1},\dots,\mathrm{head_h})W^O$$

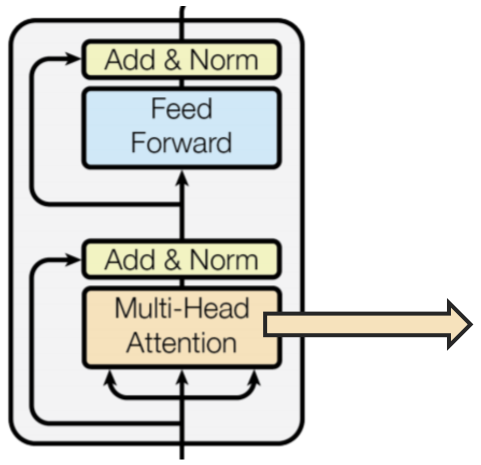


$$Embedding\ Dimension = d \cdot h$$

Transformer Layer Overview

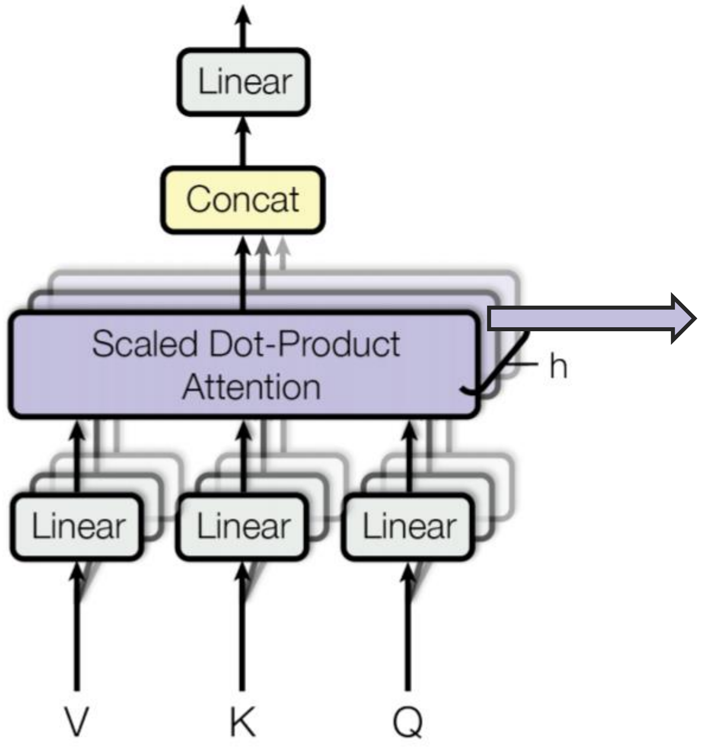
Transformer Layer

(there are N layers)

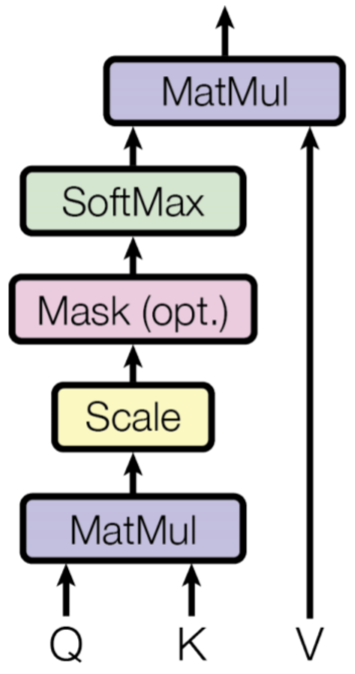


Multi-Head Attention

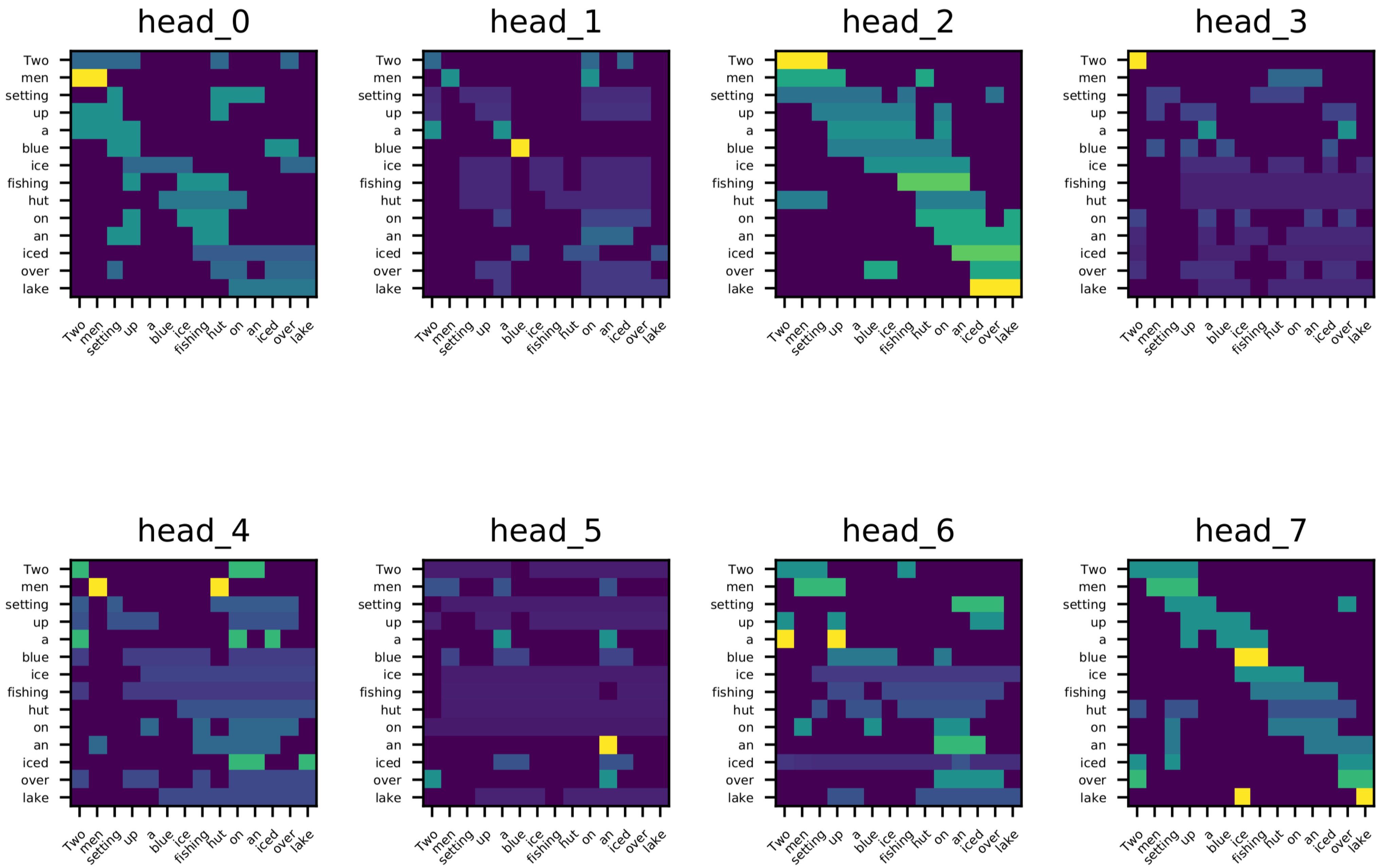
(there are h heads)



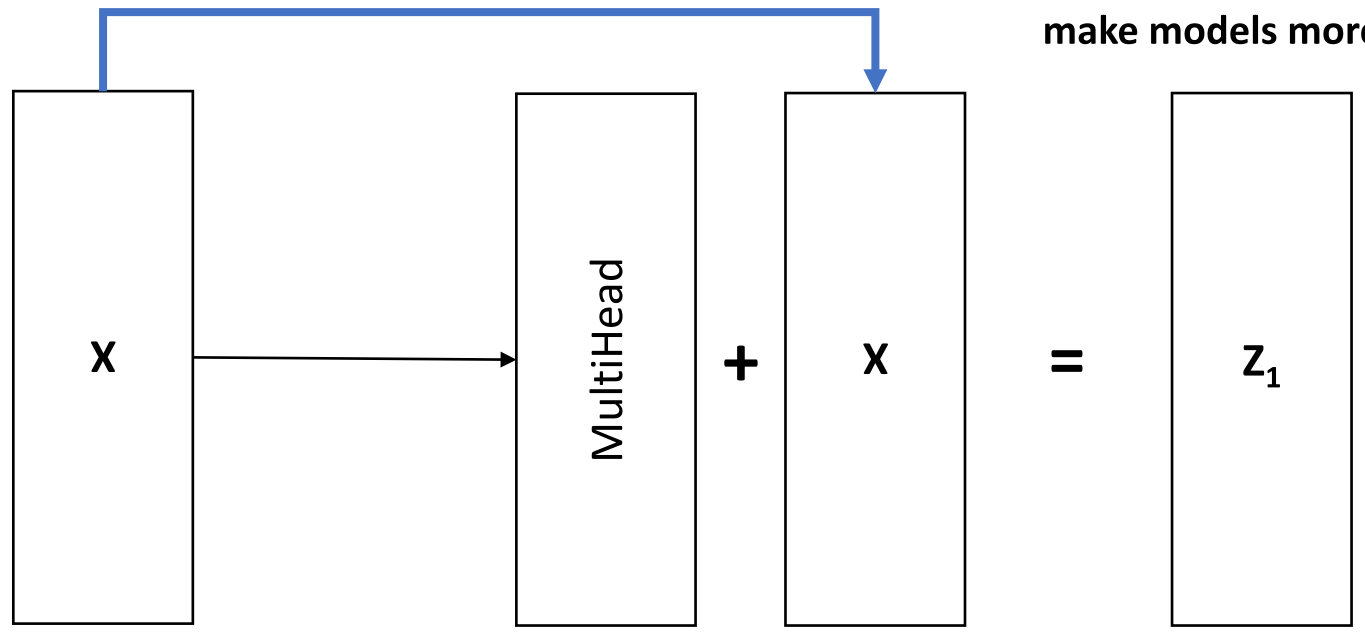
Single-Head Attention



epoch_6_enc_self_attn

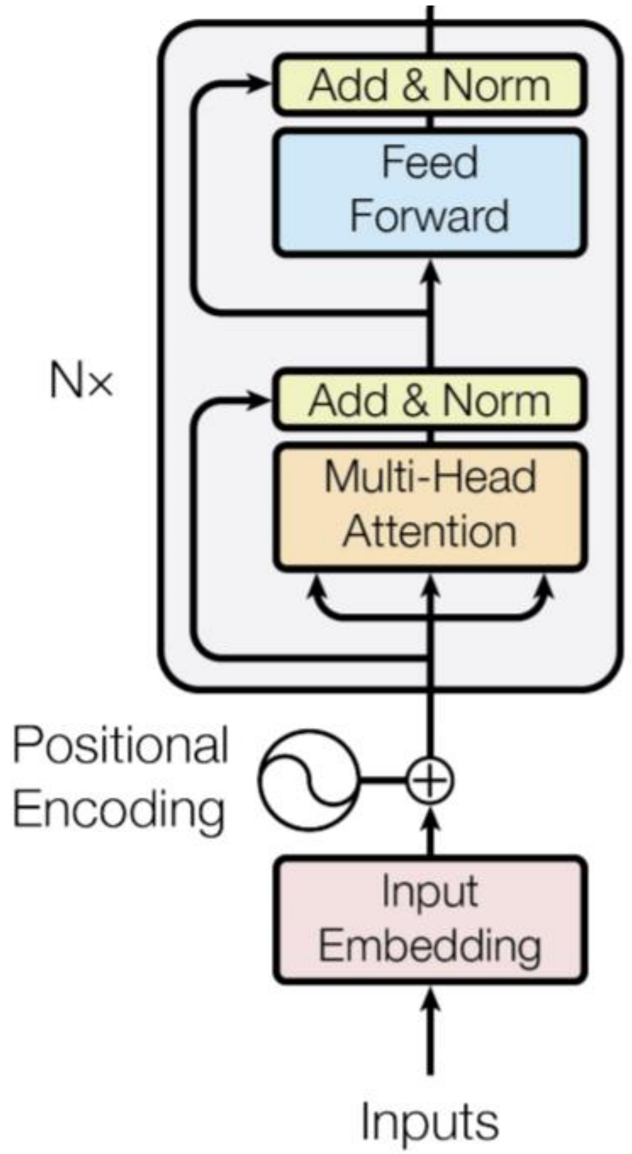


Residual, add and norm



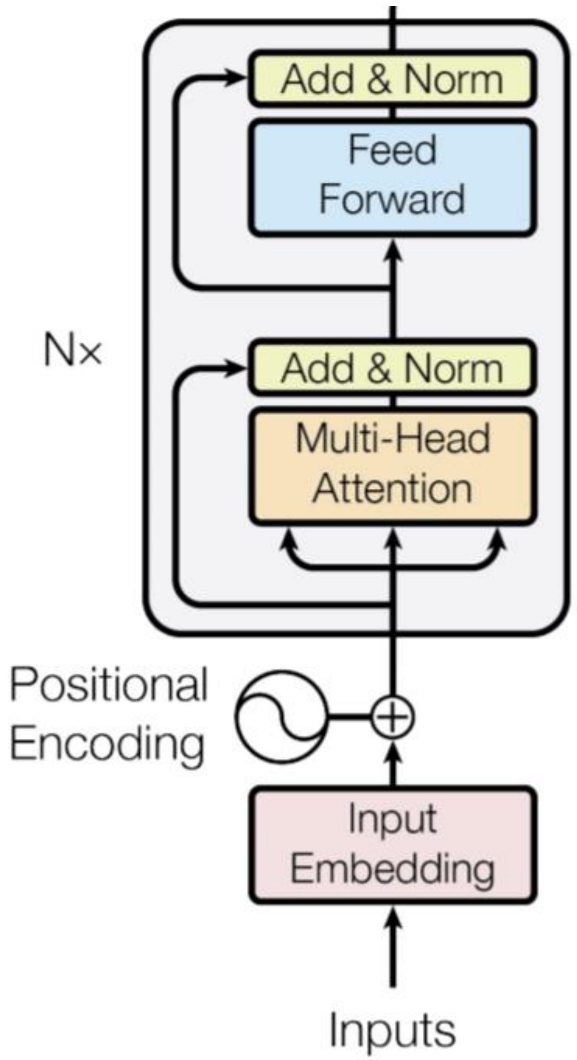
Residual connection helps preserving positional information and the original embeddings

Layer normalization help models train faster and make models more stable.



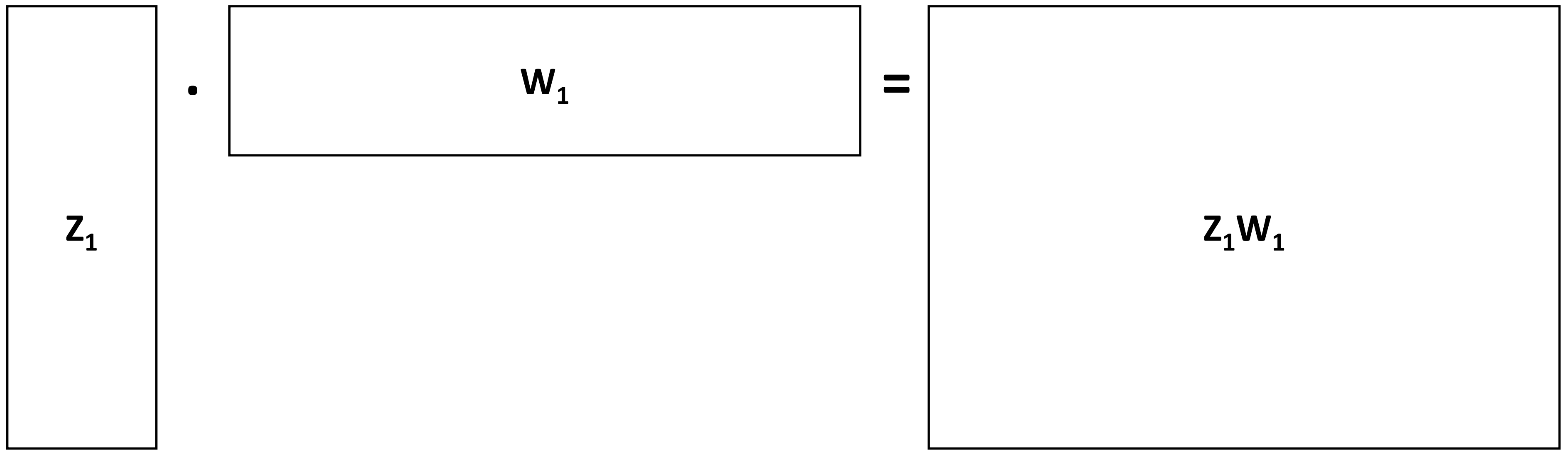
Feed forward layer

- The FF layer allows modeling non-linear relations



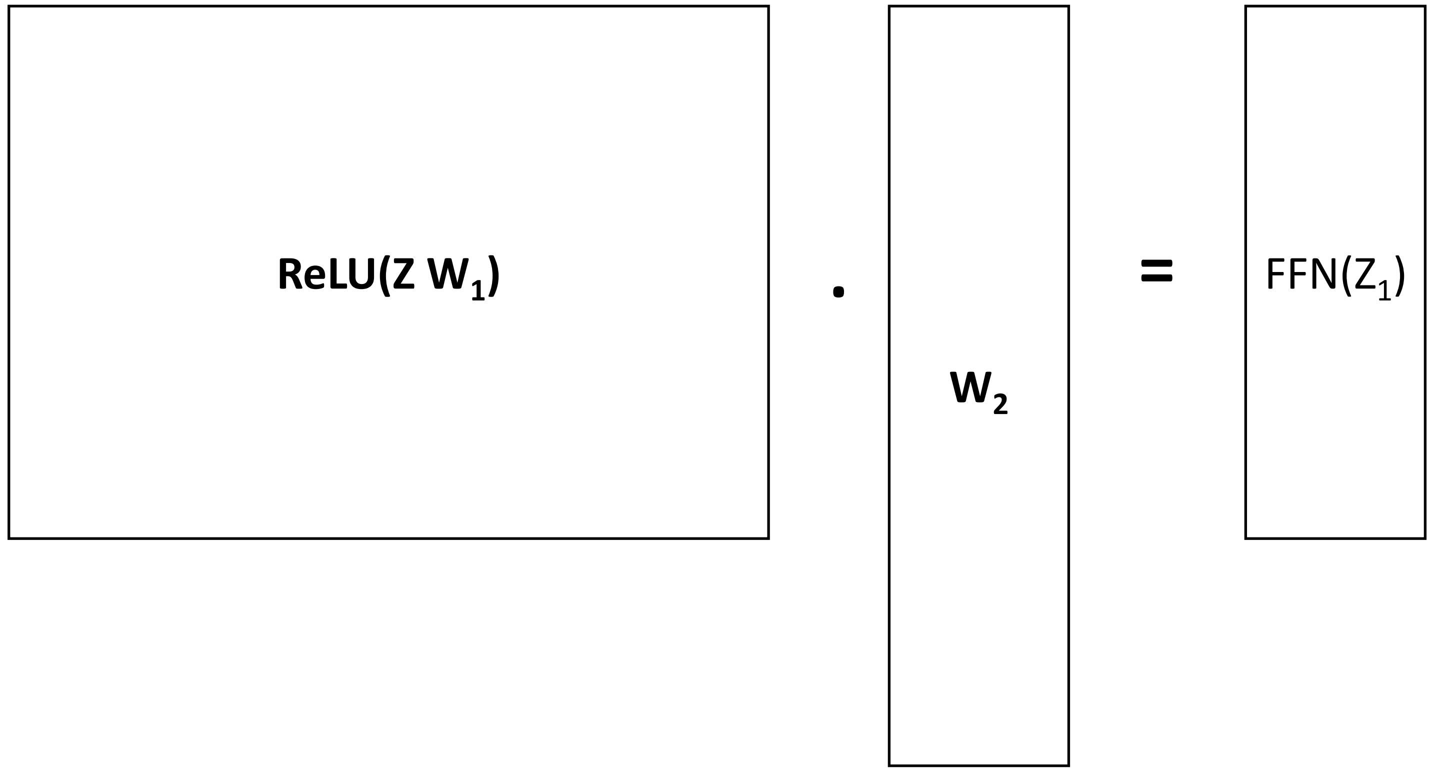
Feed forward RELU layer

$$\mathrm{FFN}(x) = \max(0, \boxed{xW_1} + b_1)W_2 + b_2$$



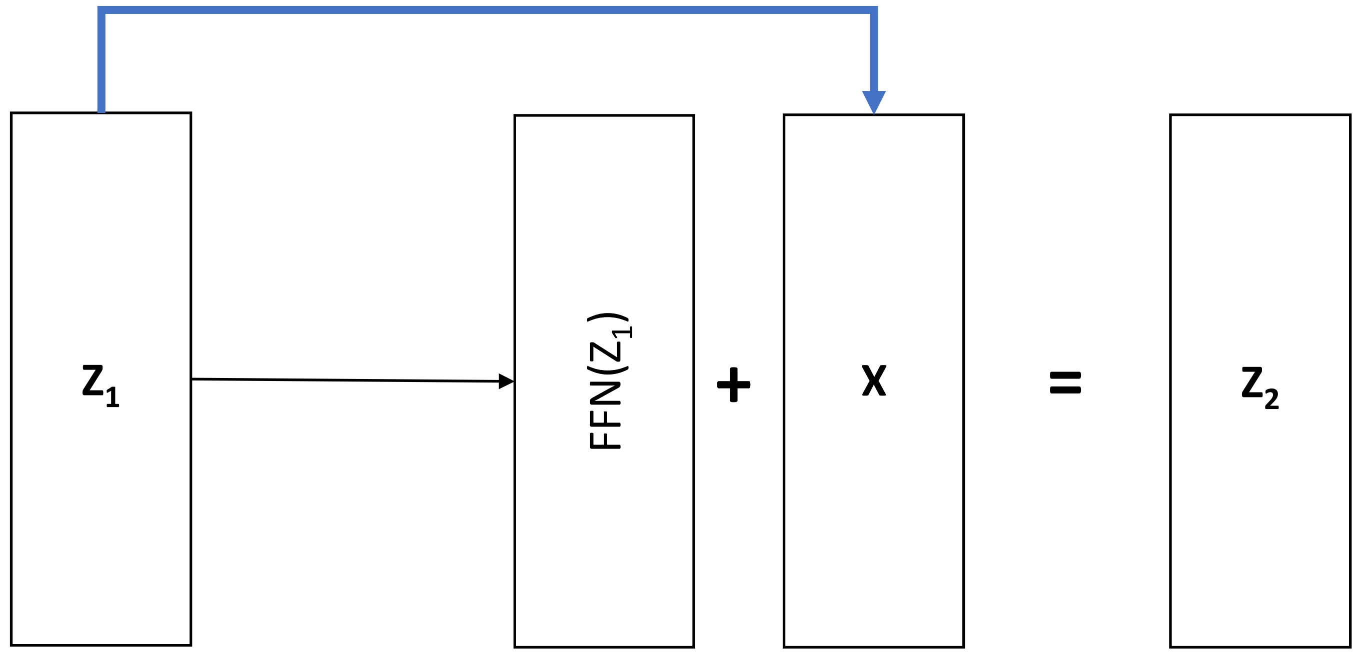
Feed forward

$$\mathrm{FFN}(x) = \boxed{\max}(0, xW_1 + b_1\boxed{)W_2} + b_2$$



Add and normalization layer

Residual connection helps preserving positional information and the original embeddings

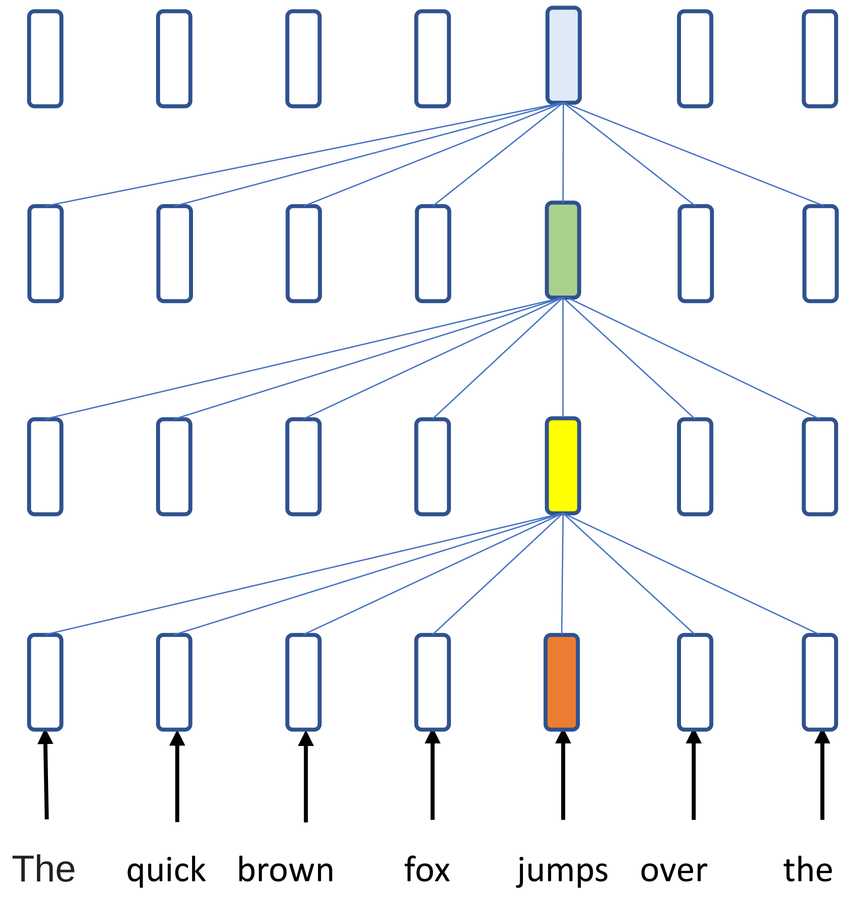


Layer normalization is a trick to help models train faster.

cut down on uninformative variation in hidden vector values by normalizing to unit mean and standard deviation within each layer.

Mutiple layers to transform the embeddings

- The input embeddings have
  - the global token embedding vectors
  - Positional embedding
- Each layer will capture the word interactions within the same sentence
- As we go up in the number of layers, more abstract interactions are captured.



The Transformer

- Position representations:
  - Specify the sequence order, since self-attention is an unordered function of its inputs.
- Self-attention:
  - The basis of the method.
- Nonlinearities:
  - At the output of the self-attention block
  - Frequently implemented as a simple feed-forward network.
- Masking:
  - In order to parallelize operations while not looking at particular parts of the sequence.

Summary

- Self-attention mechanisms
- Transformer (encoder only)
- Contextual embeddings
- Readings:
  - Dan Jurafsky and James H. Martin, Speech and Language Processing, Chapter 11 https://web.stanford.edu/~jurafsky/slp3/11.pdf