CS 5740: Natural Language Processing

Self-Attention and Transformers

Instructor: Yoav Artzi

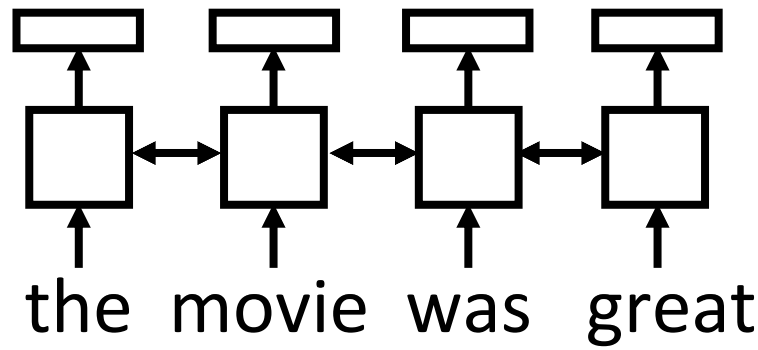
Slides adapted from Greg Durrett

Overview

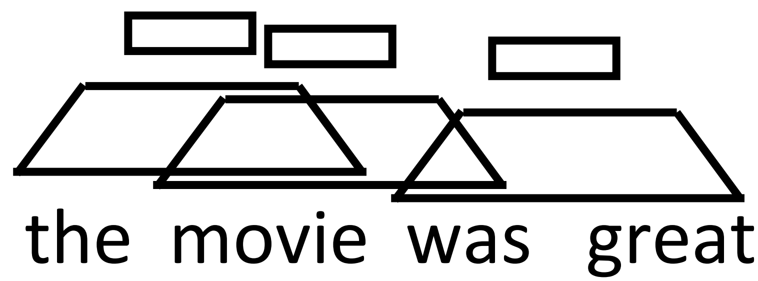
- Motivation
- Self-Attention and Transformers
- Encoder-decoder with Transformers

Encoders

- RNN: map each token vector a new context-aware token embedding using a autoregressive process



- CNN: similar outcome, but with local context using filters



- Attention can be an alternative method to generate context-dependent embeddings

LSTM/CNN Context

- What context do we want token embeddings to take into account?

The ballerina is very excited that she will dance in the show.

- What words need to be used as context here?
    - Pronouns context should be the antecedents (i.e., what they refer to)
    - Ambiguous words should consider local context
    - Words should look at syntactic parents/children
- Problem: very hard with RNNs and CNNs, even if possible

LSTM/CNN Context

- Want:

The ballerina is very excited that she will dance in the show.

- LSTMs/CNNs: tend to be local

The ballerina is very excited that she will dance in the show.

- To appropriately contextualize, need to pass information over long distances for each word

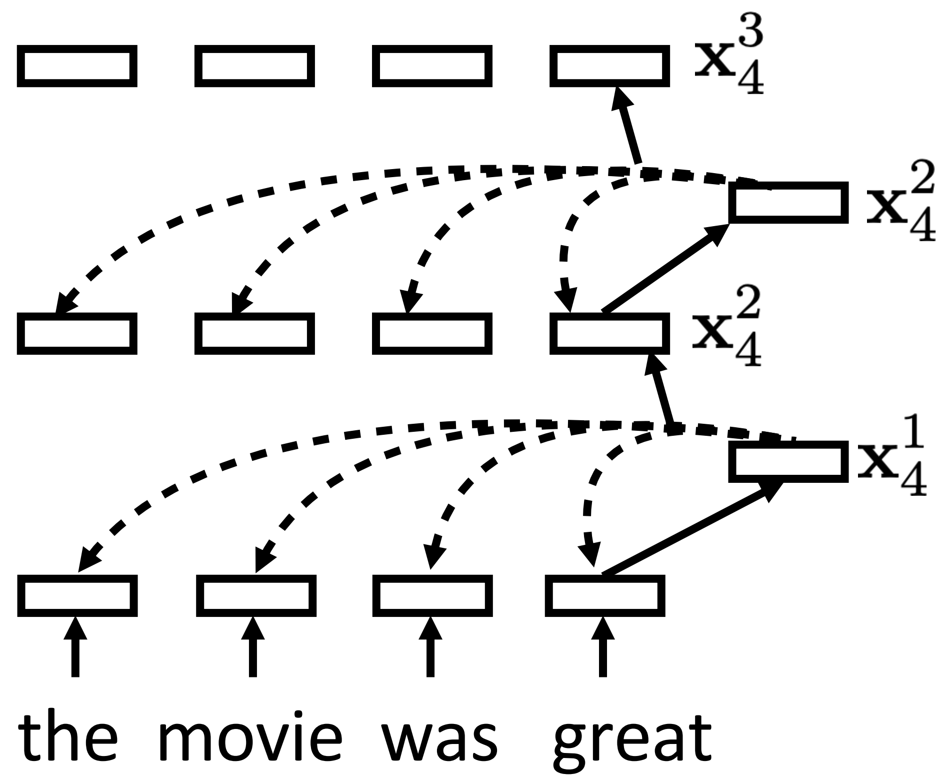
Self-attention

- Each word is a ***query*** to form attention over all tokens
- This generates a context-dependent representation of each token: a weighted sum of all tokens
- The attention weights dynamically mix how much is taken from each token
- Can run this process iteratively, at each step computing self-attention on the output of the previous level

[Vaswani et al. 2017]

Self-attention

- Each word is a ***query*** to form attention over all tokens
- This generates a context-dependent representation of each token: a weighted sum of all tokens
- The attention weights dynamically mix how much is taken from each token
- Can run this process iteratively, at each step computing self-attention on the output of the previous level



[Vaswani et al. 2017]

Self-attention w/Dot-product

$$k : \text{level number}$$

$$X : \text{input vectors}$$

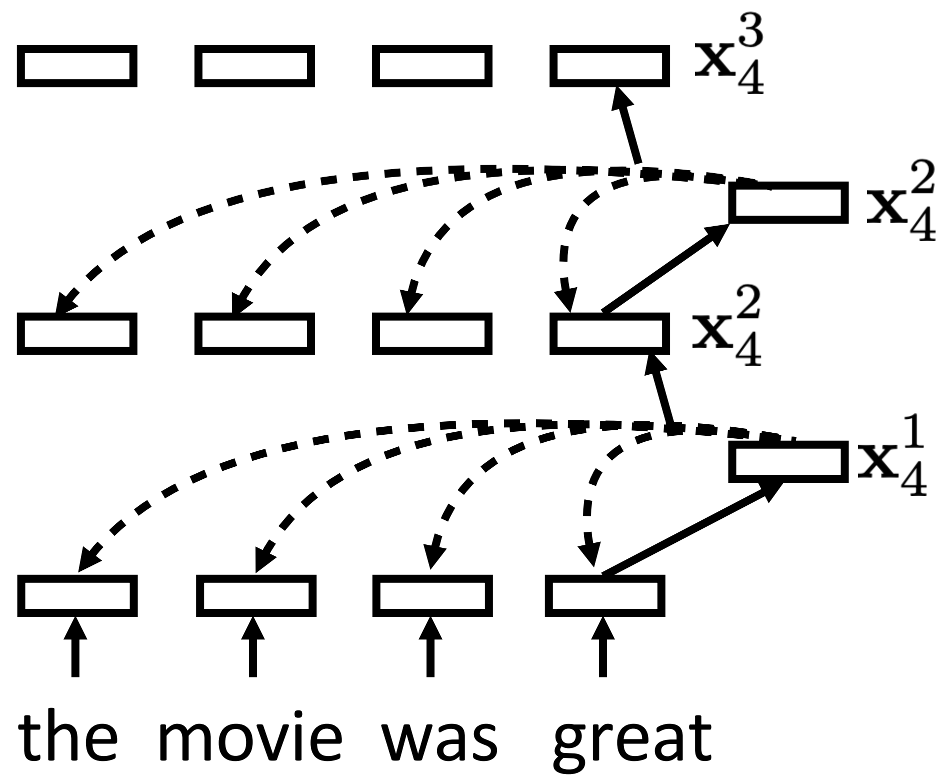
$$X = \mathbf{x}_1, \ldots, \mathbf{x}_n$$

$$\mathbf{x}_i^1 = \mathbf{x}_i$$

$$\bar{\alpha}_{i,j}^k = \mathbf{x}_i^{k-1} \cdot \mathbf{x}_j^{k-1}$$

$$\alpha_i^k = \text{softmax}(\bar{\alpha}_{i,1}^k, \ldots, \bar{\alpha}_{i,n}^k)$$

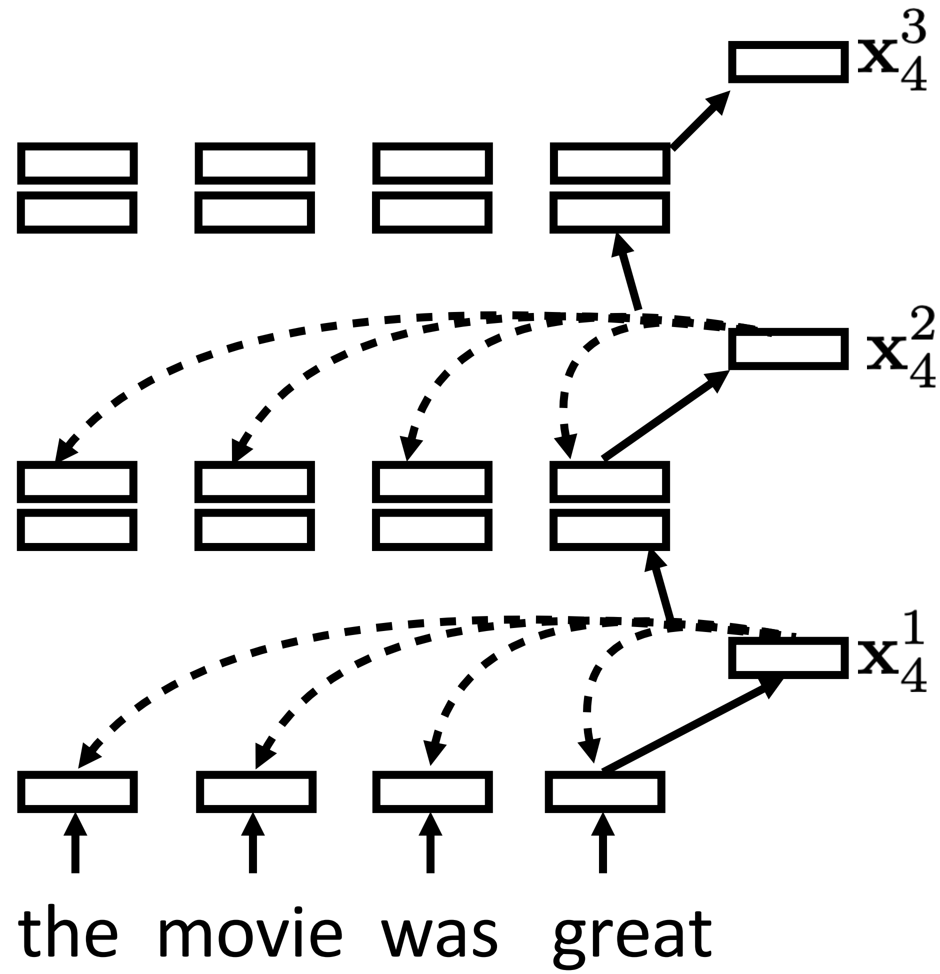
$$\mathbf{x}_i^k = \sum_{i=1}^{n} \alpha_{i,j}^k \mathbf{x}_j^{k-1}$$



[Vaswani et al. 2017]

Multiple Attention Heads

- Multiple attention heads can learn to attend in different ways
- Why multiple heads? Softmax operations often end up peaky, making it hard to put weight on multiple items
- Requires additional parameters to compute different attention values and transform vectors
- Analogous to multiple convolutional filters



Multiple Attention Heads

k : level number

L : number of heads

X : input vectors

$$X = \mathbf{x}_1, \ldots, \mathbf{x}_n$$

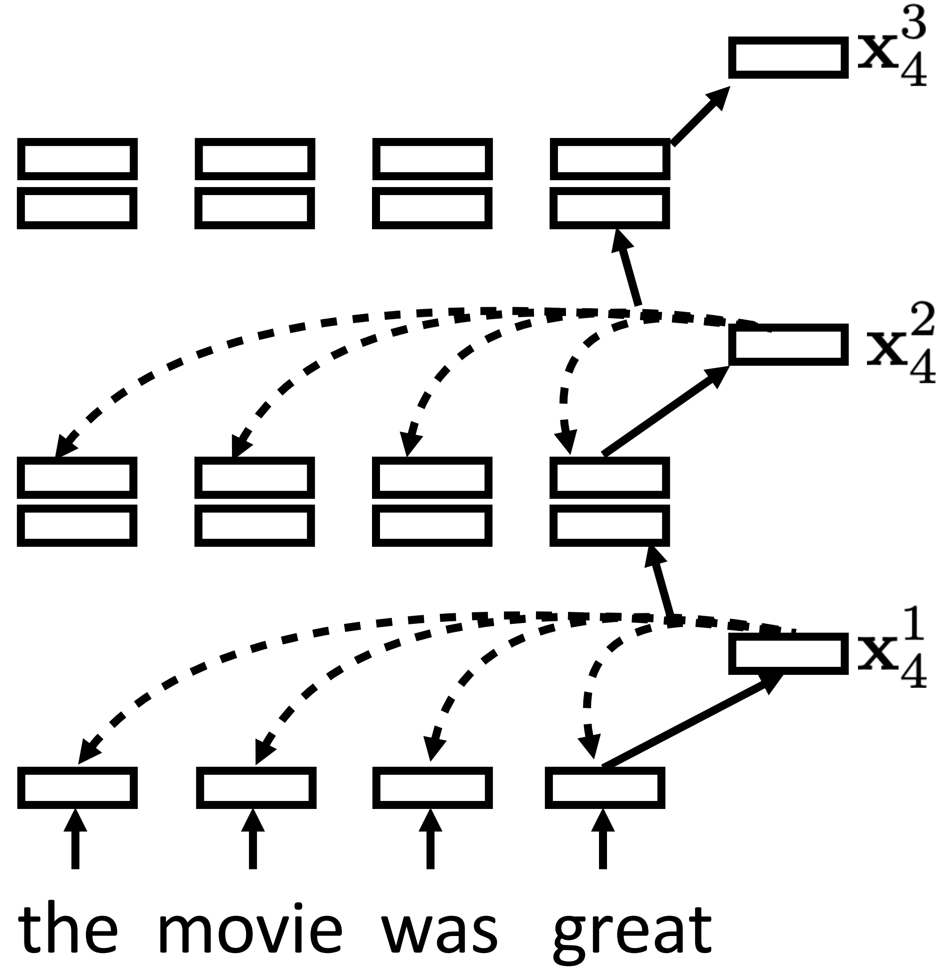
$$\mathbf{x}_i^1 = \mathbf{x}_i$$

$$\bar{\alpha}_{i,j}^{k,l} = \mathbf{x}_i^{k-1}\mathbf{Q}^{k,l} \cdot \mathbf{x}_j^{k-1}\mathbf{K}^{k,l}$$

$$\alpha_i^{k,l} = \text{softmax}(\bar{\alpha}_{i,1}^{k,l}, \ldots, \bar{\alpha}_{i,n}^{k,l})$$

$$\mathbf{x}_i^{k,l} = \sum_{i=1}^{n} \alpha_{i,j}^{k,l} \mathbf{x}_j^{k-1} \mathbf{V}^{k,l}$$

$$\mathbf{x}_i^k = [\mathbf{x}_i^{k,1}; \ldots; \mathbf{x}_i^{k,L}]$$



What Can Self-attention do?

The ballerina is very excited that she will dance in the show.

0	0.5	0	0	0.1	0.1	0	0.1	0.2	0	0	0
0	0.1	0	0	0	0	0	0	0.5	0	0.4	0

- Attend to nearby related terms
- But just the same to far semantically related terms

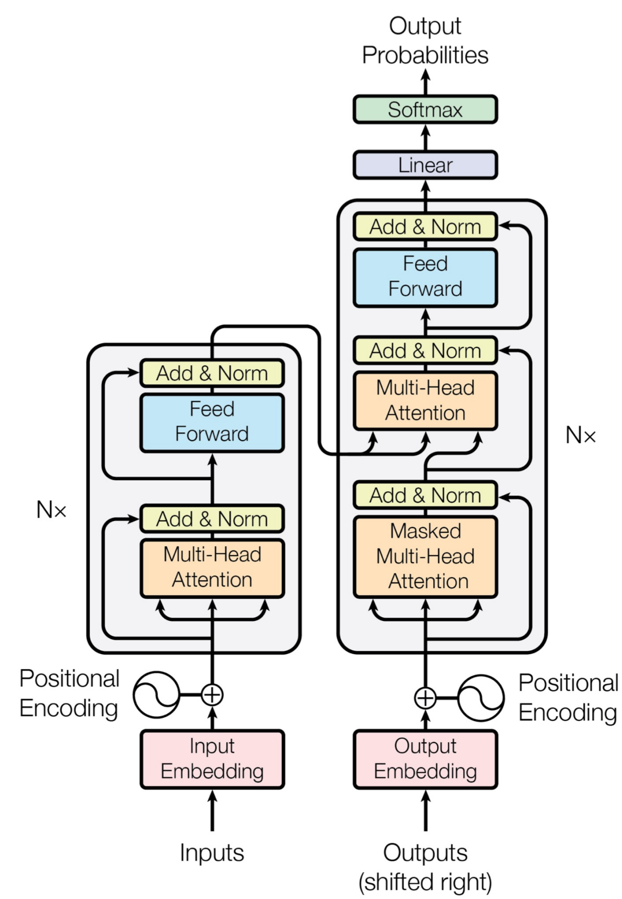
Details Details Details

- Self-attention is the basic building block of an architecture called Transformers

- Many details to get it to work

- Significant improvements for many tasks, starting with machine translation (Vaswani et al. 2017) and later context-dependent pre-trained embeddings (BERT; Devlin et al. 2018)

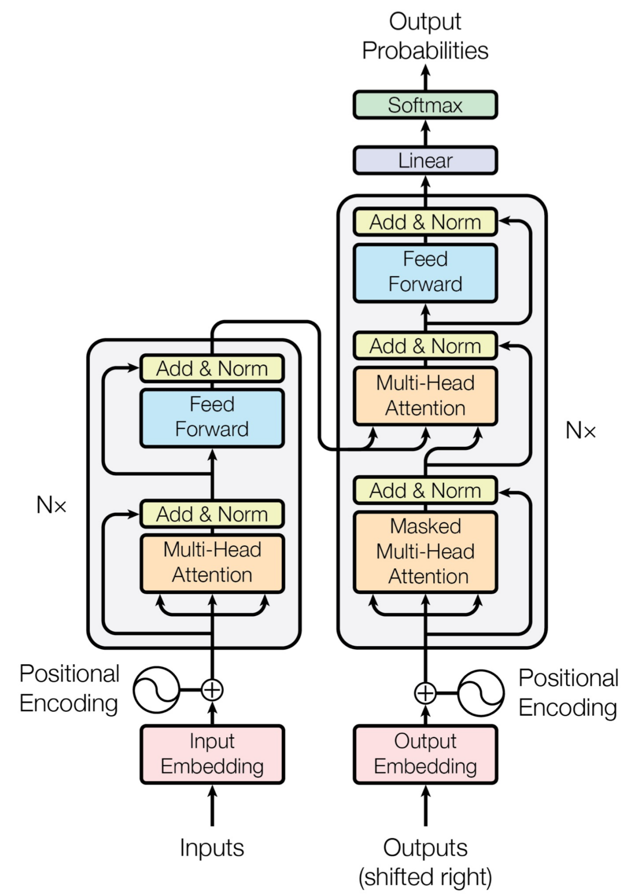
- A detailed technical description (with code): https://www.aclweb.org/anthology/W18-2509/



[Figure from Vaswani et al. 2017]

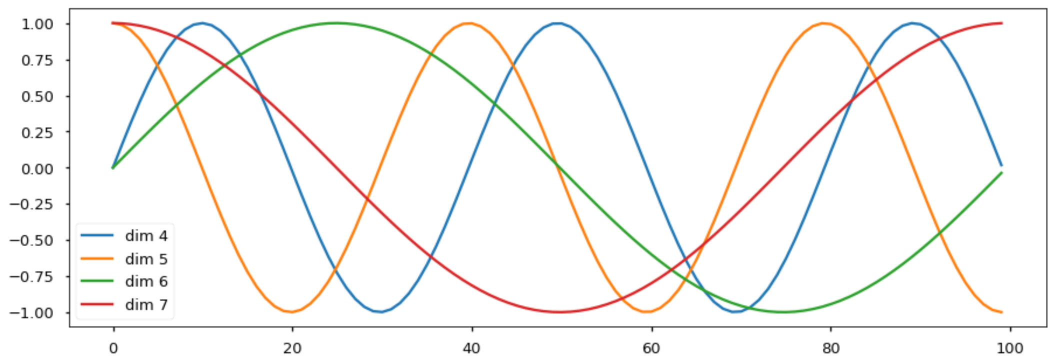
MT with Transformers

- Input: sentence in source language
- Output: sentence in target language
- Encoder Transformer processes the input
- Decoder Transformer generates the output
- More generally: this defines an encoder-decoder architecture with Transformers

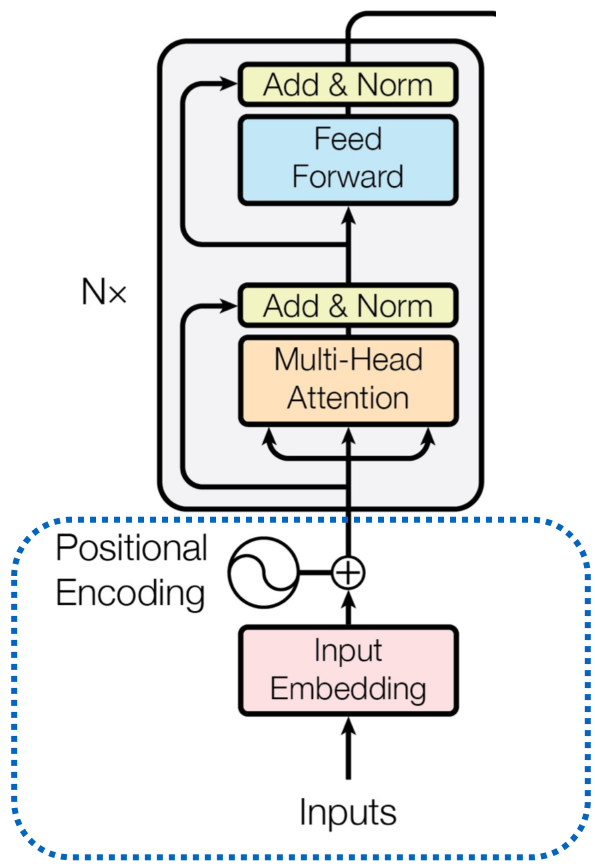


Encoder

- Self-attention is not order-sensitive
- Need to add positional information
- Add time-dependent function to token embeddings (sin and cos)



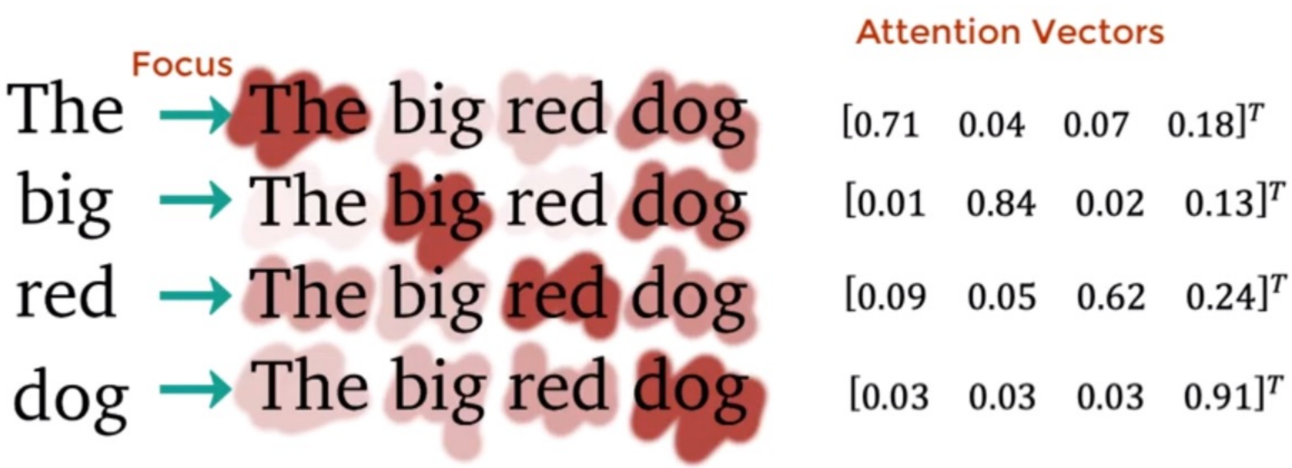
- Output: a set of token embeddings

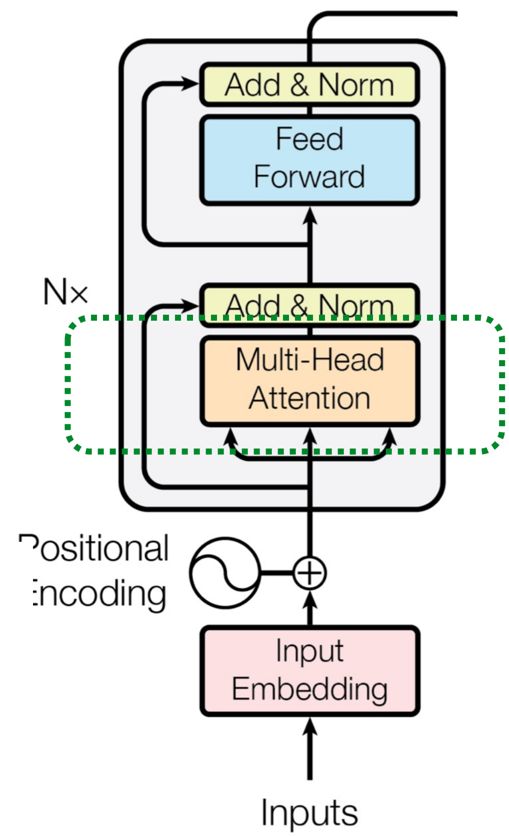


[Positional Dimensions figure from Rush 2018]

Encoder

- Use parameterized attention
- Multiple attention heads, each with separate parameters
- This increases the attention flexibility





Decoder

- Can't attend to the whole output
- Why? It doesn't exist yet!
- Tokens are generated one-by-one
- Solution: mask tokens that are not predicted yet in the attention
- First: self-attend to the output only
  Second: attend to both input and output