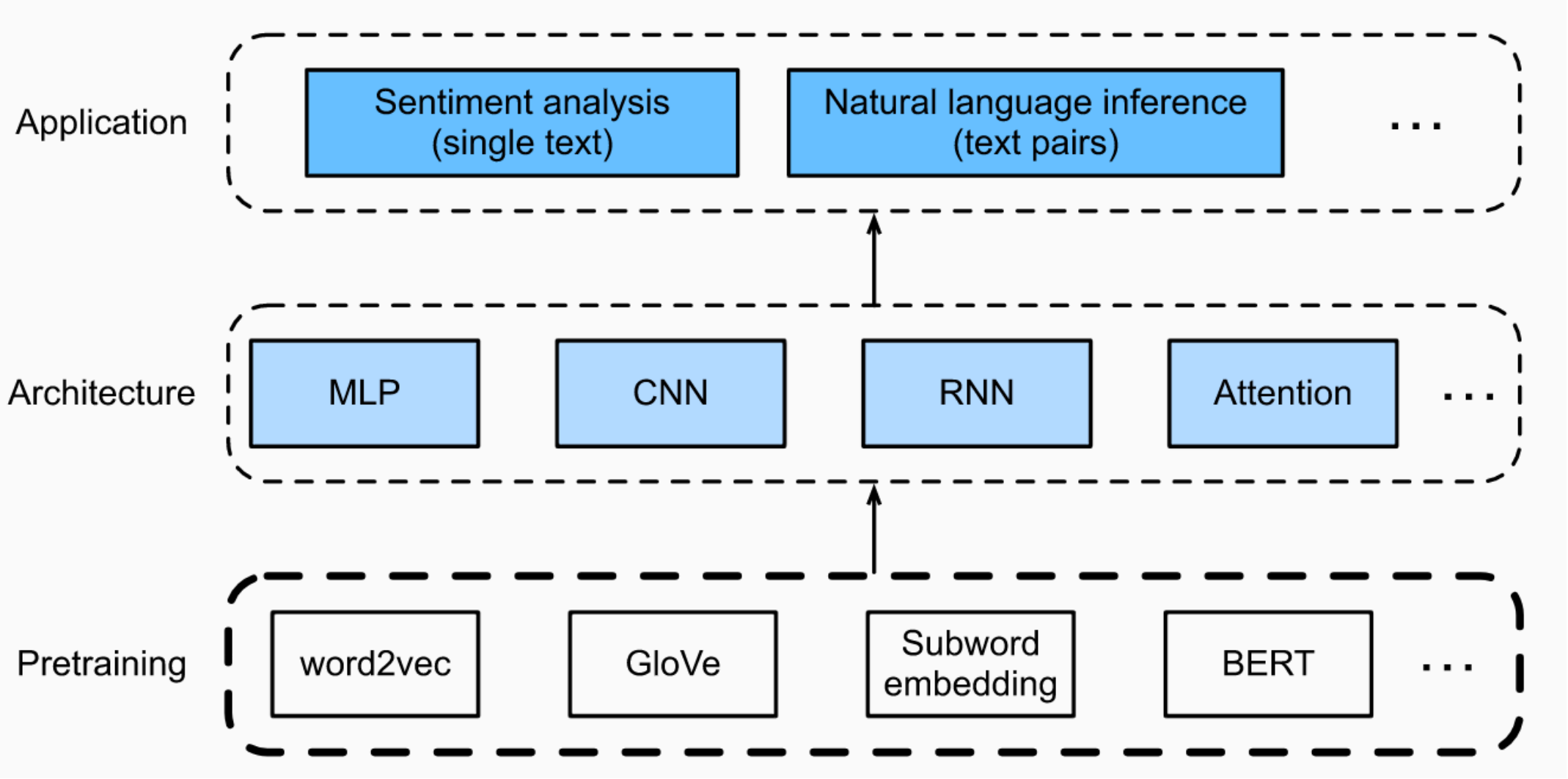
Attention and Transformers

Arsalan Mosenia (am12546@nyu.edu)

Chinmay Hegde (chinmay.h@nyu.edu)

NLP Overview



NLP Applications

- Machine Translation
- Sentiment Analysis (e.g., Social Media Monitoring)
- Question-answering (e.g., Chatbots)
- Hiring and Recruitment
- Automatic Summarization
- Spam Filtering
- Voice Assistants

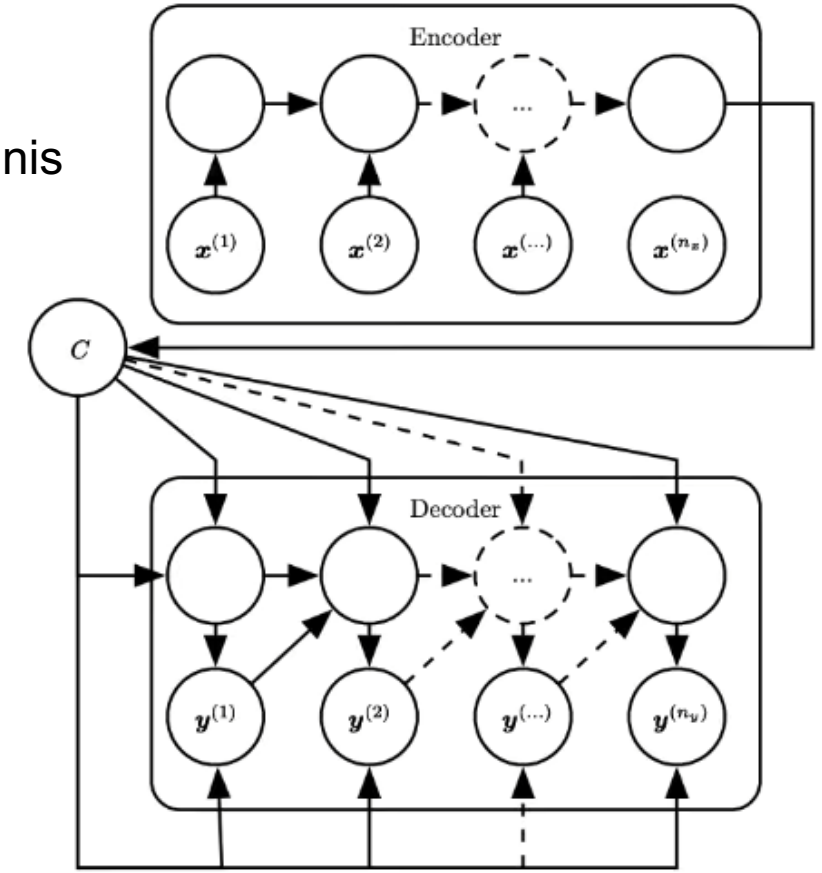
Machine Translation

- Statistical Machine Translation (SMT)
- Rule-based Machine Translation (RBMT)
- Hybrid Machine Translation (HMT)
- Neural Machine Translation (NMT)

Classic Sequence-to-Sequence Problem

Decoder-Encoder Model

Les pauvres sont démunis



The poor do not have any money

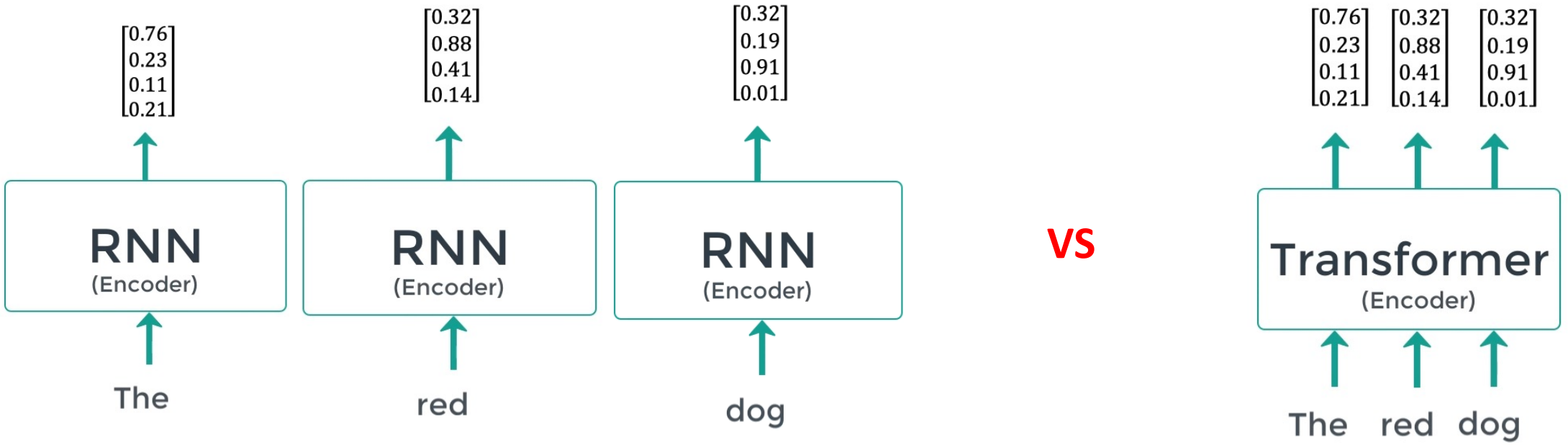
RNN Challenges

RNNs

1. Long-range Dependencies
2. Gradient Vanishing/Exploding
3. Large #Training Steps
4. Recurrence prevents Parallel computation

Transformers

1. Facilitate Long-range dependencies
2. No Gradient Vanishing/Exploding
3. Fewer #Training Steps
4. Parallelization



Attention Mechanism

Mimics *Retrieval Process* in a database:

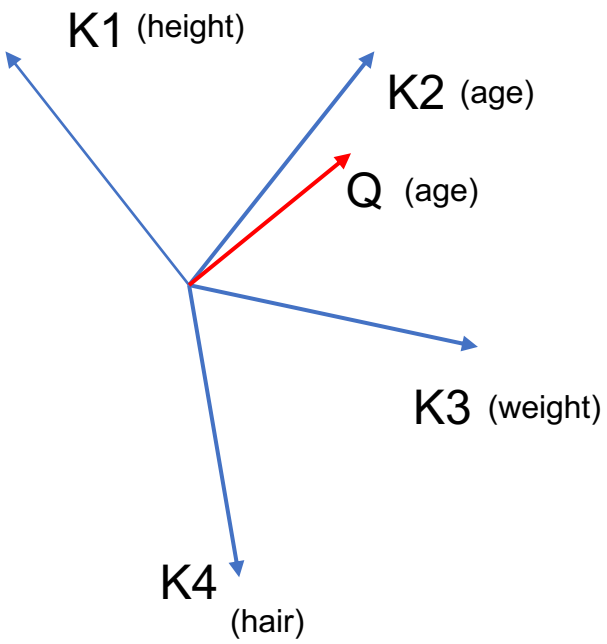
- Query
- Key
- Value

$$Attention(Query, Key, Value) = \sum_i Similarity(Query, Key_i) * Value_i$$

$Query$

Key	Value
Key 1	Val1
Key 2	Val2
Key 3	Val3
Key 4	Val4

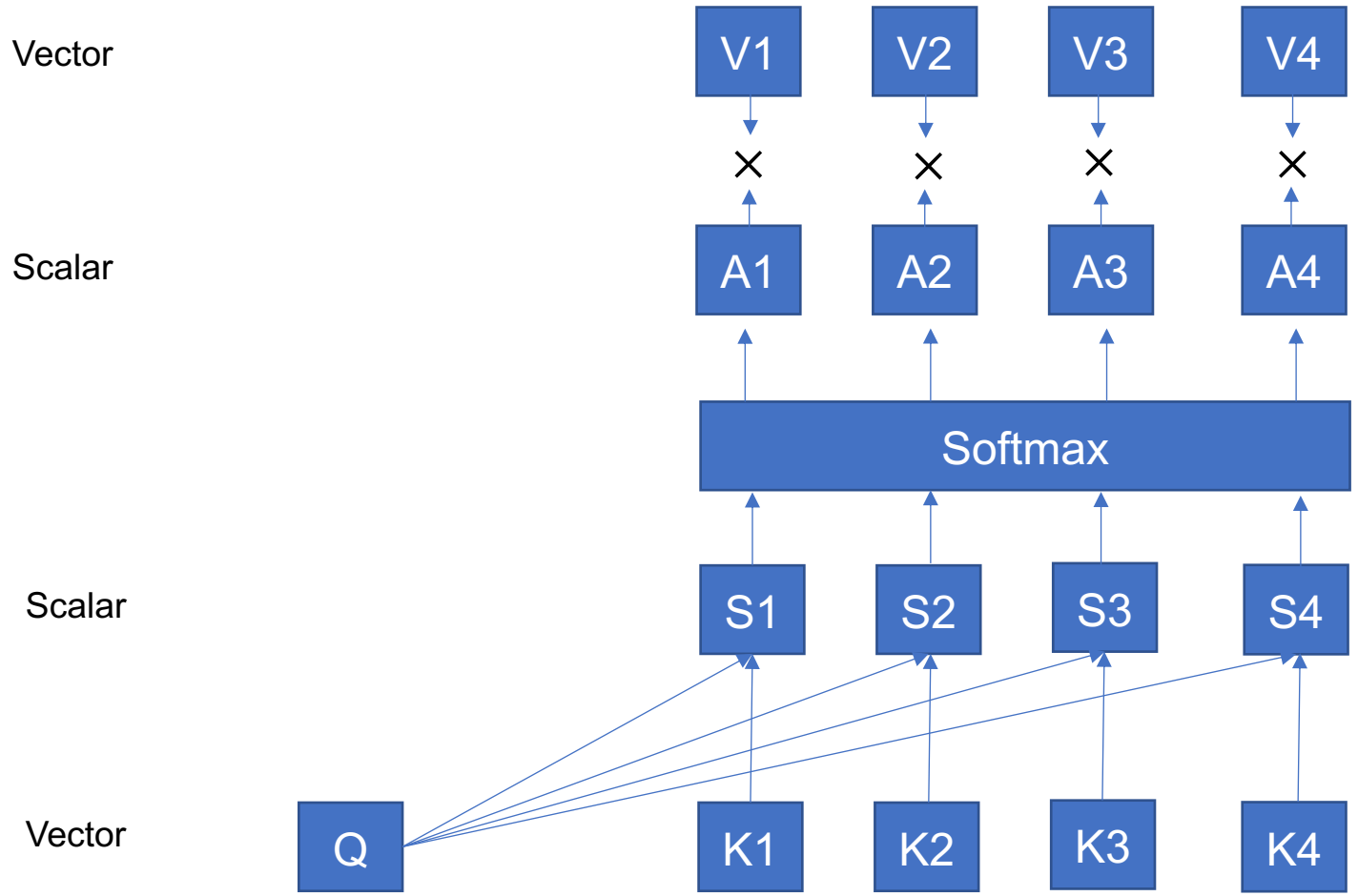
Attention Mechanism



<Q,K>	V
0.05	171
0.8	23
0.05	130
0.1	1

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

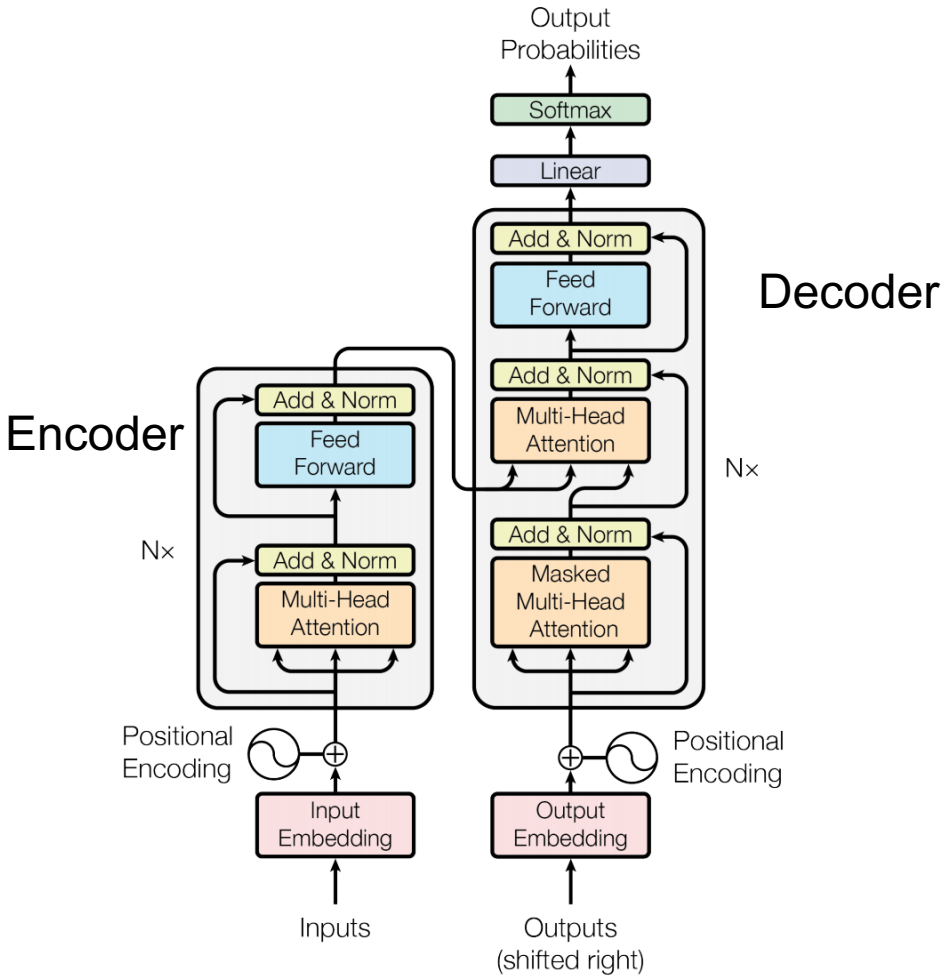
Attention Mechanism

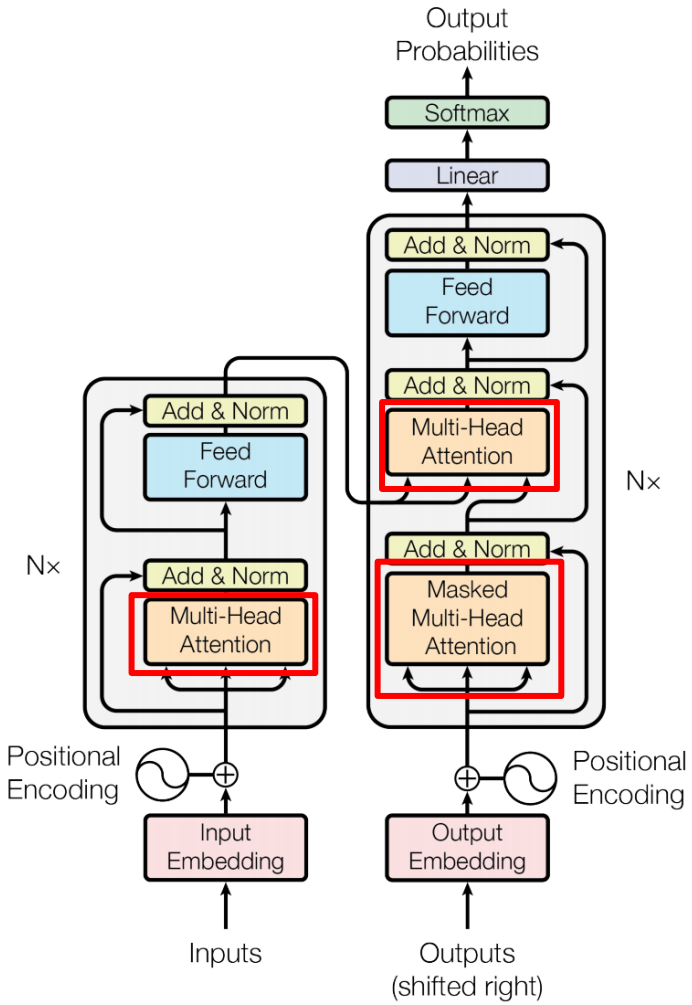


$$Attention = \sum_i A_i * V_i$$

$$S = \begin{matrix} Q^T K_i \\ Q^T K_i / \sqrt{\mathrm{d}} \\ Q^T W K_i \end{matrix}$$

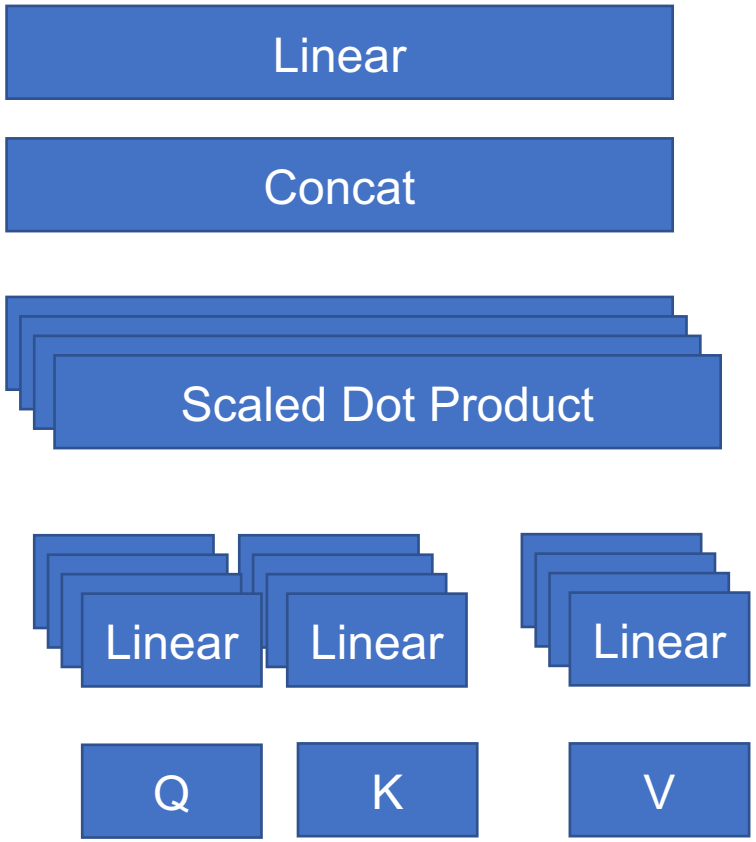
Transformer Architecture



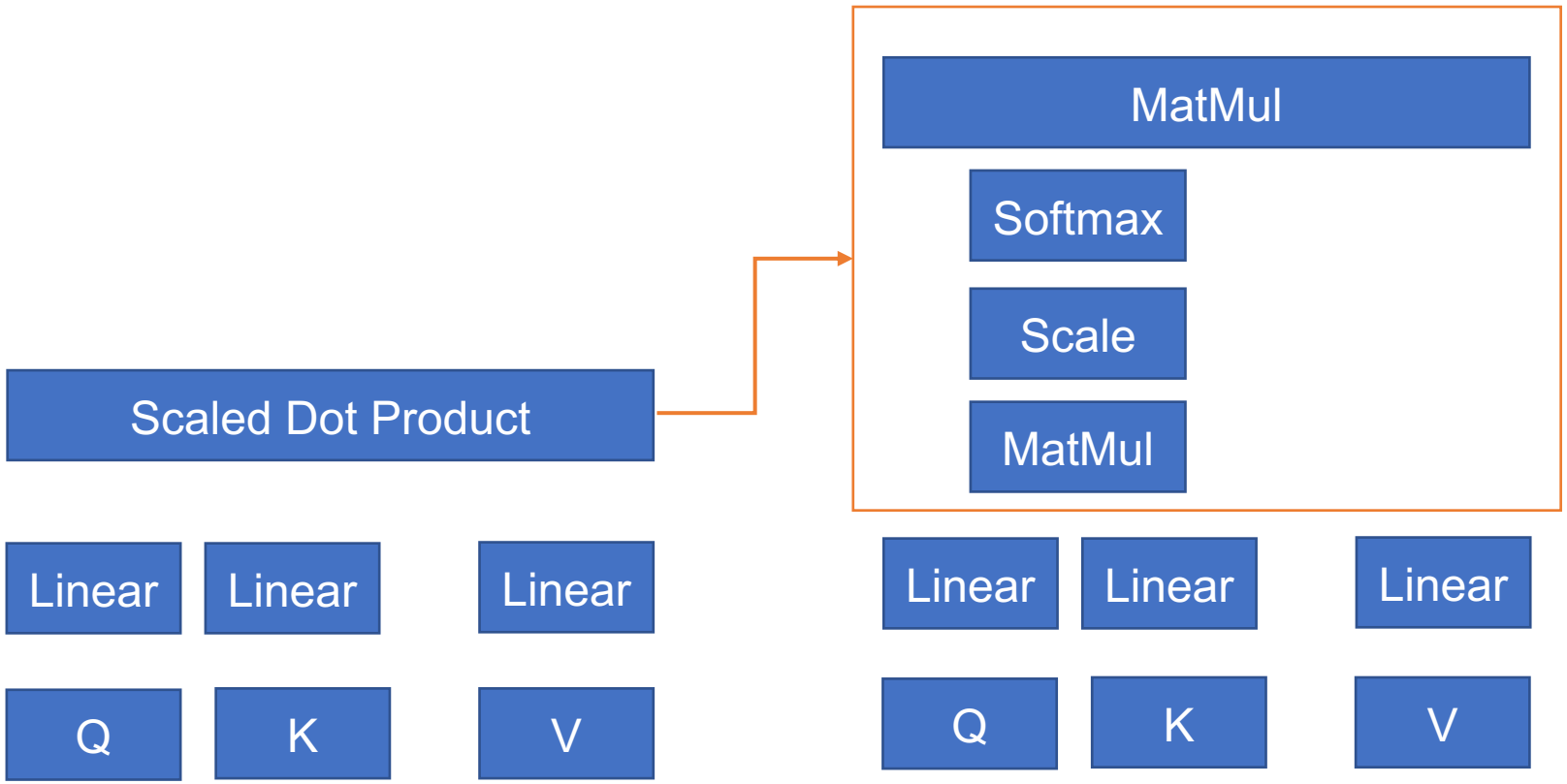


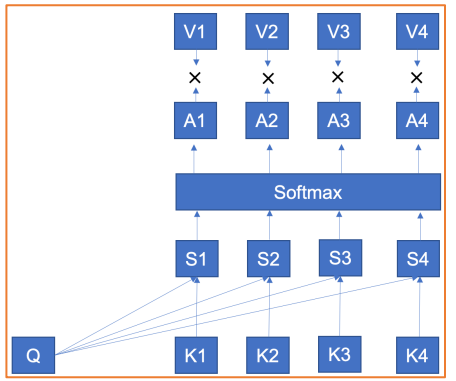
Multi-Head Attention

Multi-Head Attention

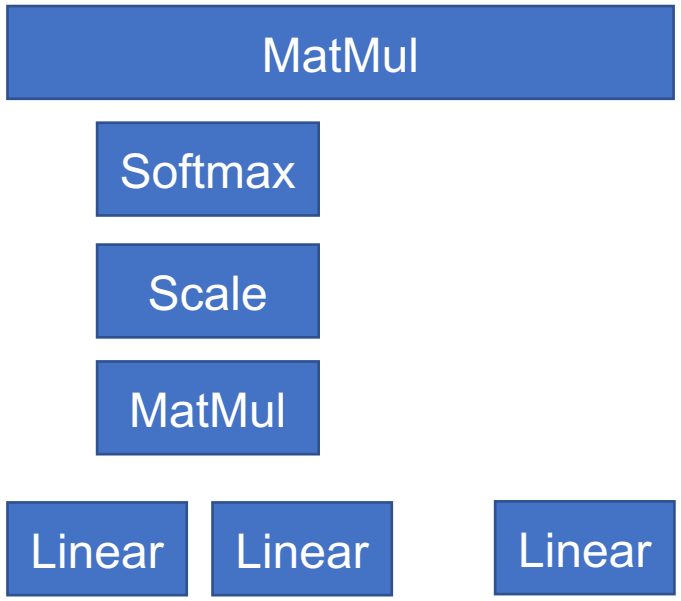


Multi-Head Attention



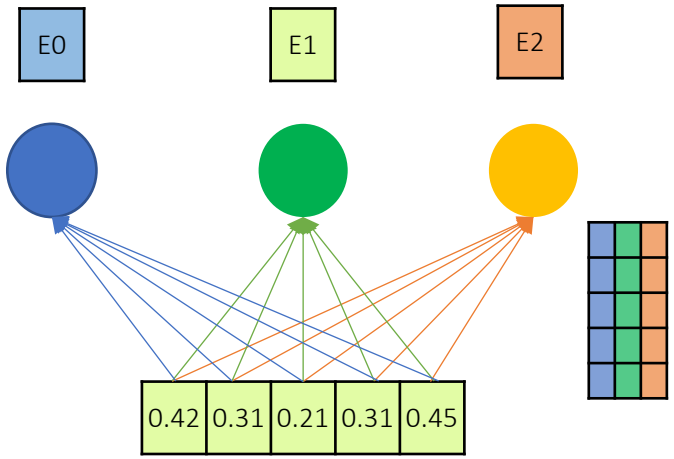


Self-Attention Mechanism

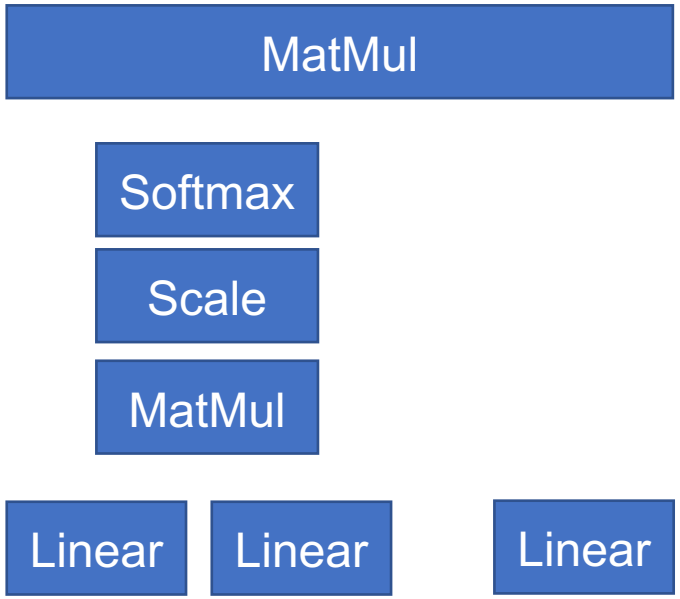


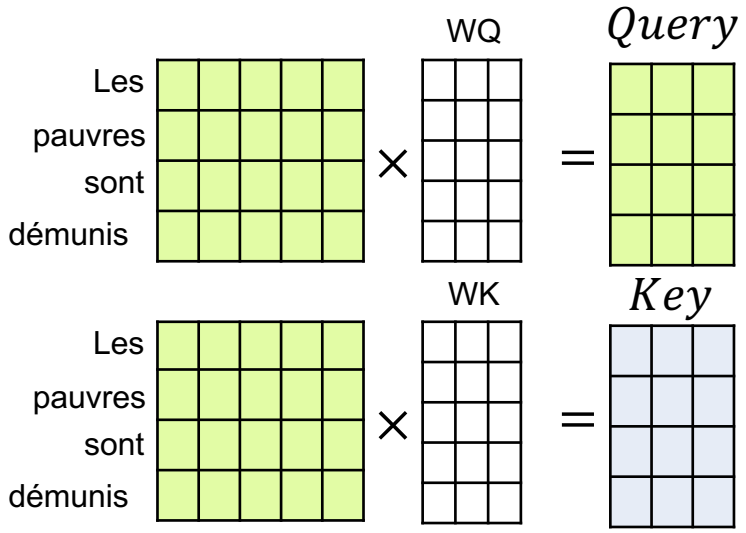
Linear Layer:

- Linear combination of elements in embeddings
- Fully-connected layer without activation = Projection
- Controlling dimensions of inputs



Self-Attention

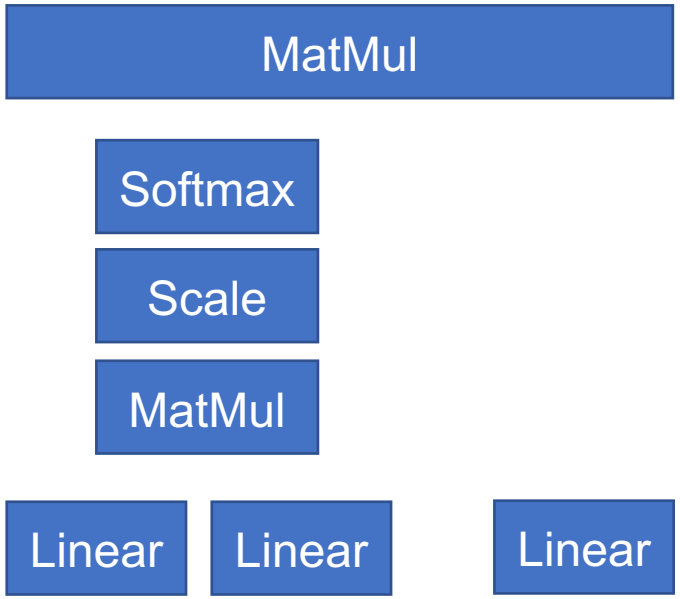


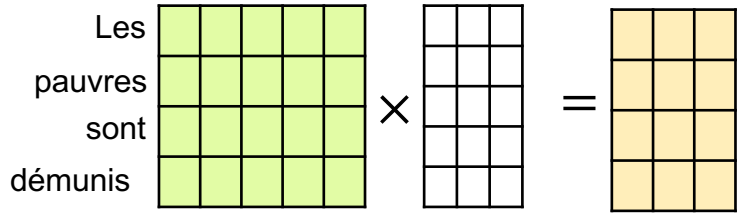


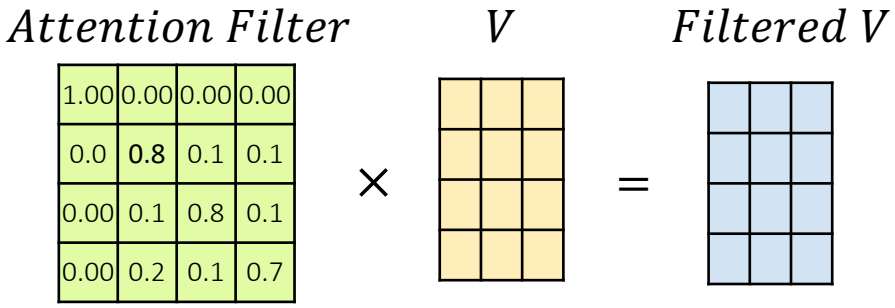
Attention Filter

1.00	0.00	0.00	0.00
0.0	0.8	0.1	0.1
0.00	0.1	0.8	0.1
0.00	0.2	0.1	0.7

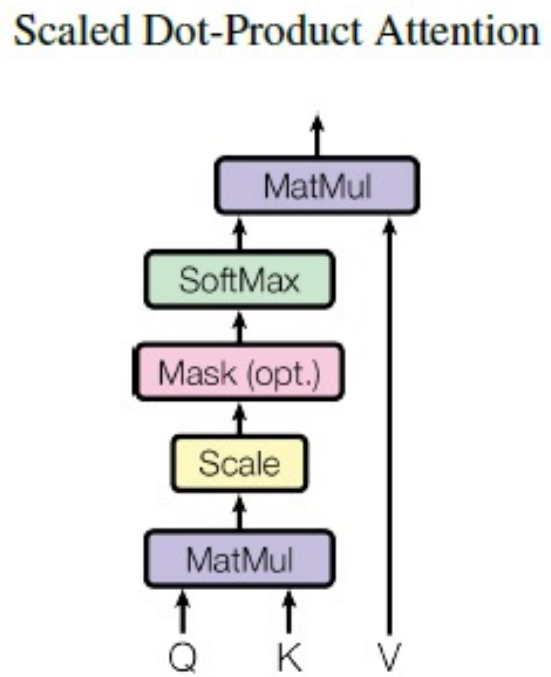
Self-Attention

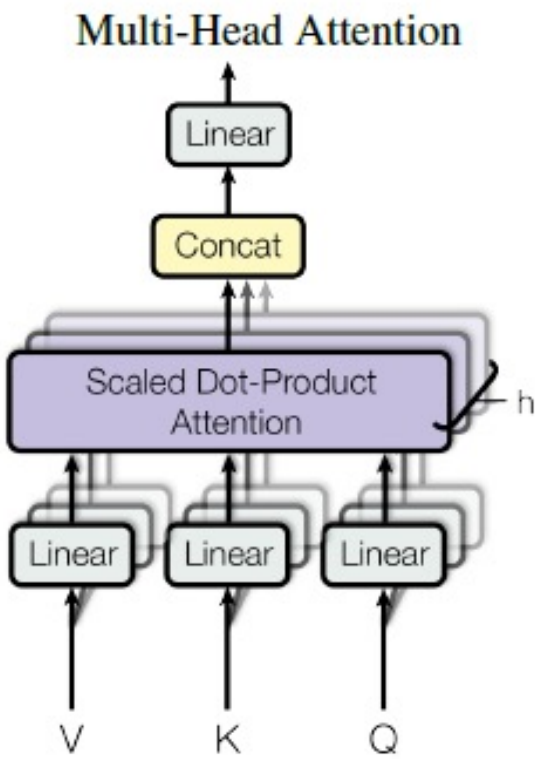






Multi-Head Attention: Summary

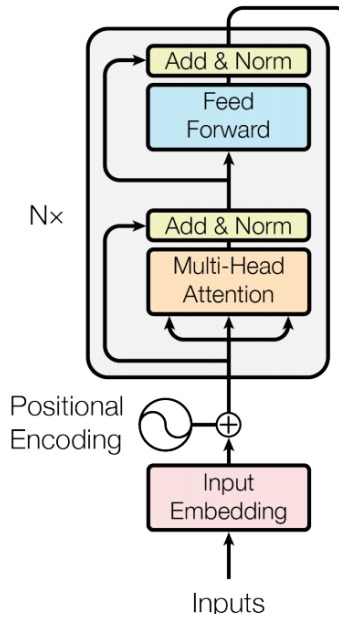




Multi-Head Attention: Intuition

The Encoder:

- Combines pairs of words, pairs of pairs of words, …
- Get the entire sentence
  - Get all words in the input sequence
  - Uses Positional Encoding
- Multiple heads: Similar to the idea of multiple filters in CNN



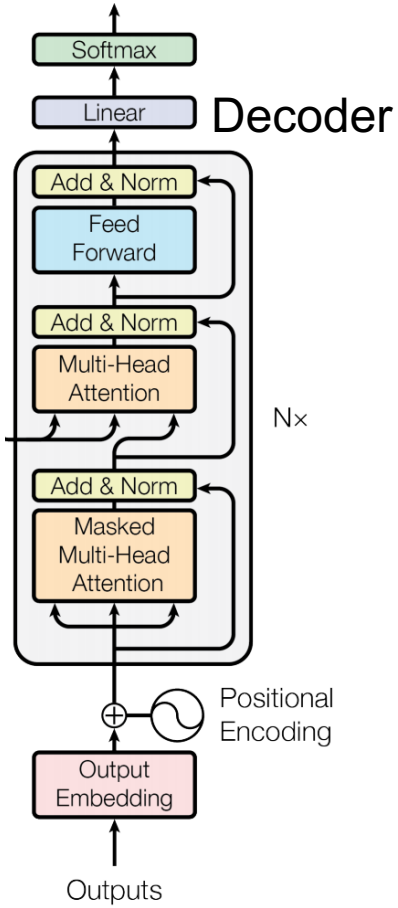
Masked Multi-Head Attention

Some of the values should be masked so that we do not use them to make combinations

In Language Translation, when we produce a word:

1. It is fine depend on previously-generated words since we generate them sequentially
2. We cannot depend on the future words since they are not generated yet

Solution: In the decoder, we need to prevent building attention based on future words



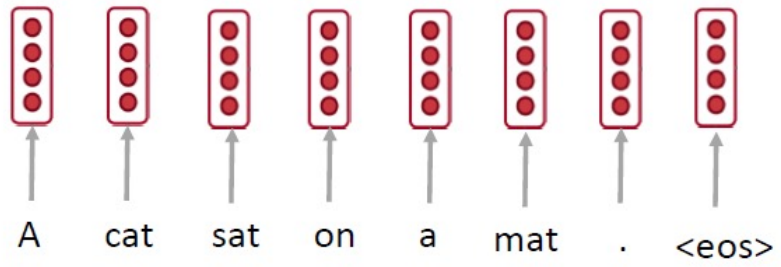
$$Attention(Q, K, V) = Softmax\left(\frac{Q^T K}{\sqrt{d}}\right)V$$

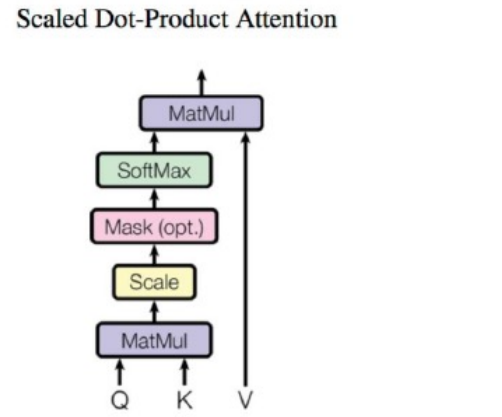
$$MaskedAttention(Q, K, V) = Softmax\left(\frac{Q^T K + M}{\sqrt{d}}\right)V$$

$$M\ Mask\ Matrix\ of\ 0s\ and\ -\infty$$

Self-Attention: Running Example

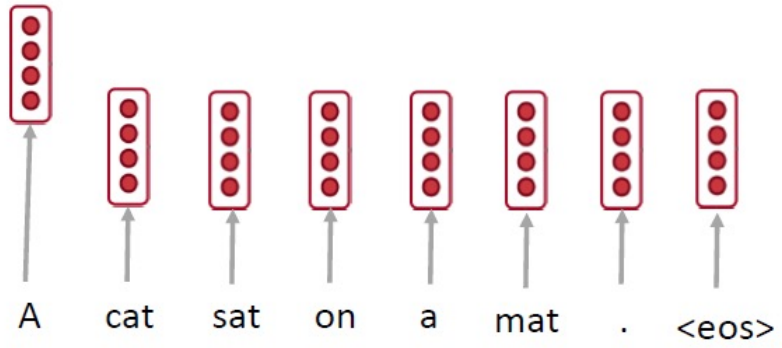
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

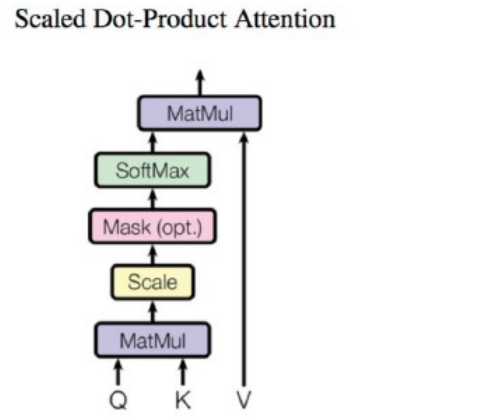




Self-Attention: Running Example

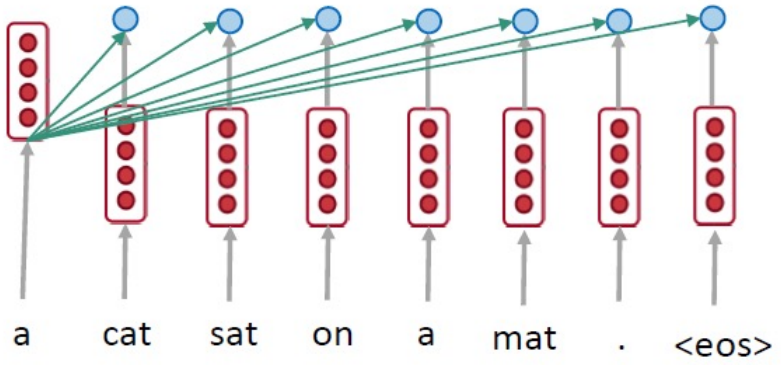
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

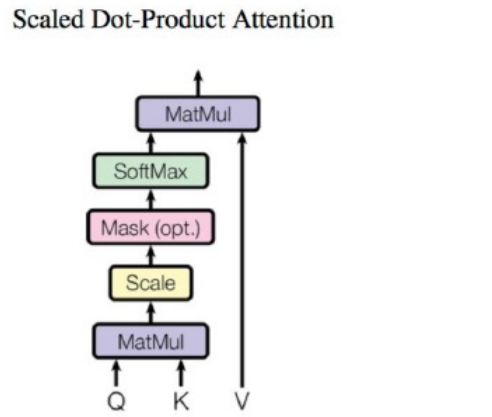




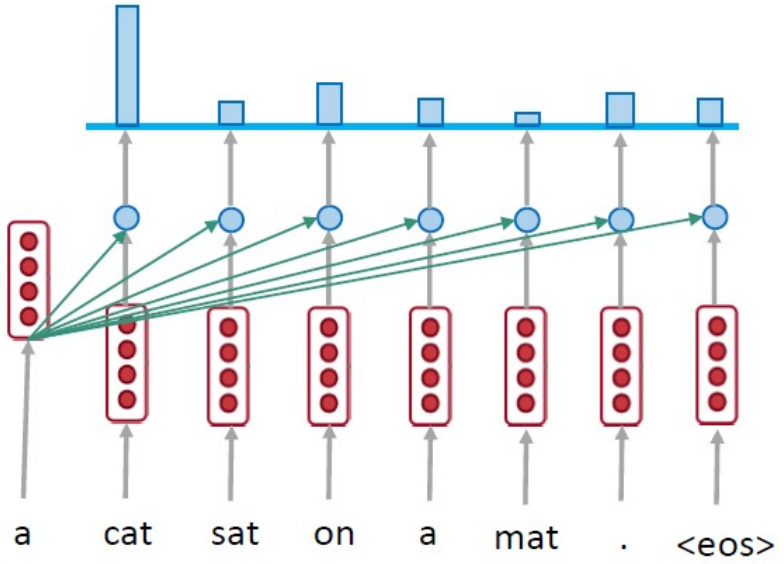
Self-Attention: Running Example

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

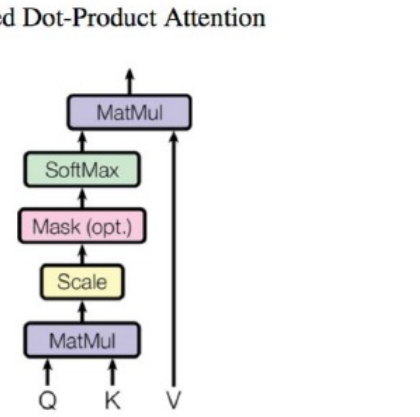




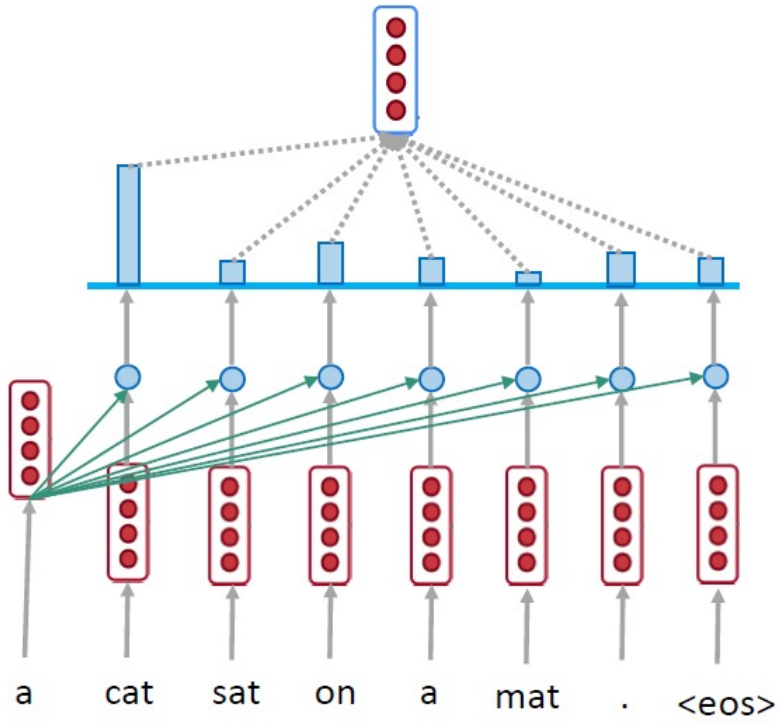
Self-Attention: Running Example



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

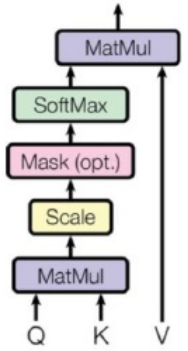


Self-Attention: Running Example

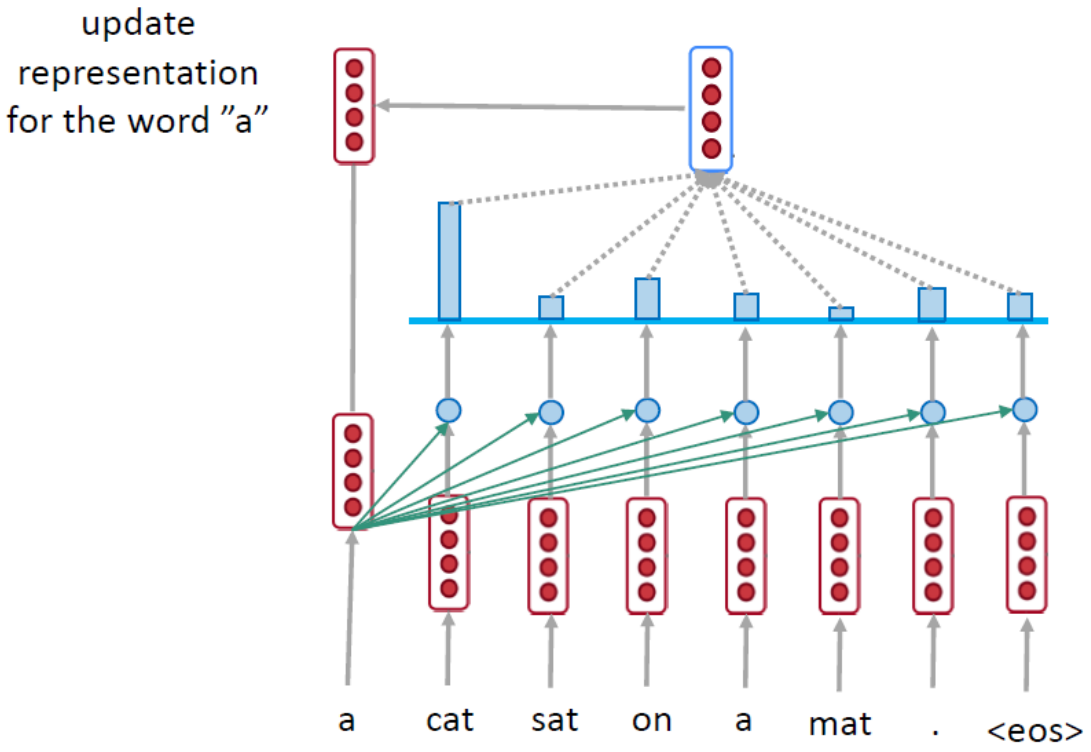


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

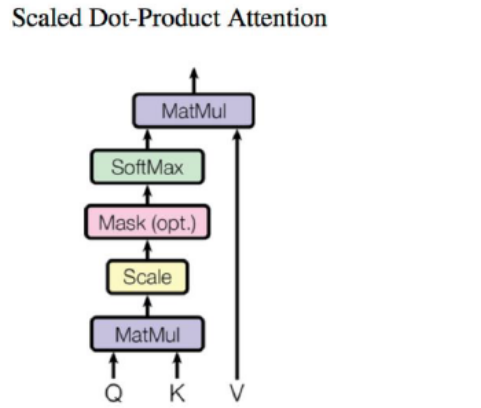
Scaled Dot-Product Attention



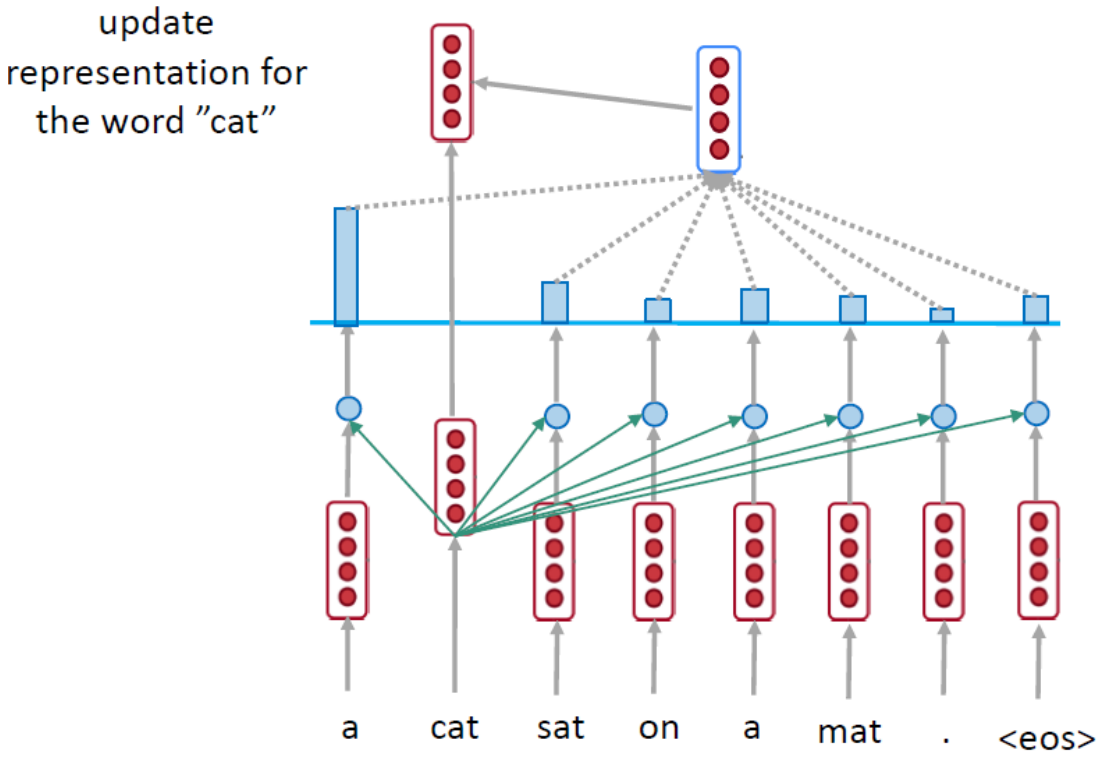
Self-Attention: Running Example



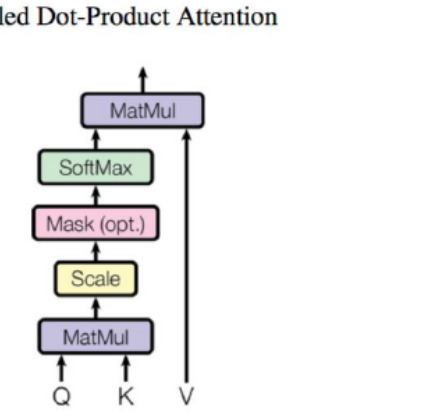
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



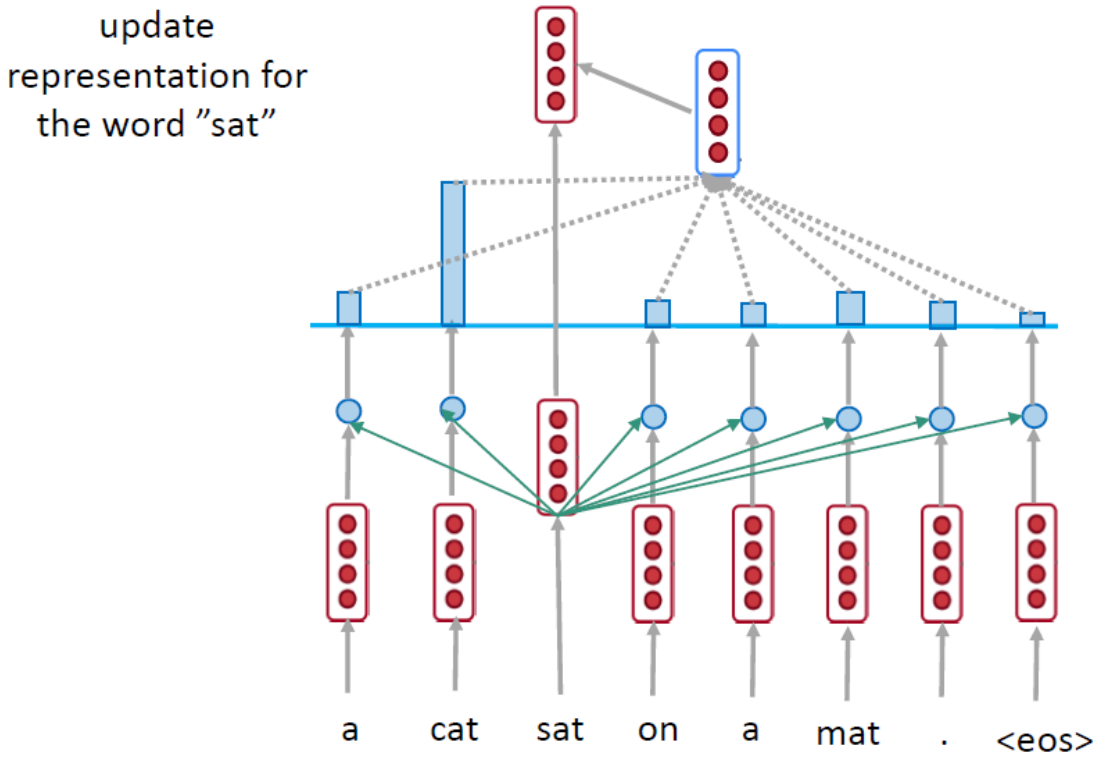
Self-Attention: Running Example



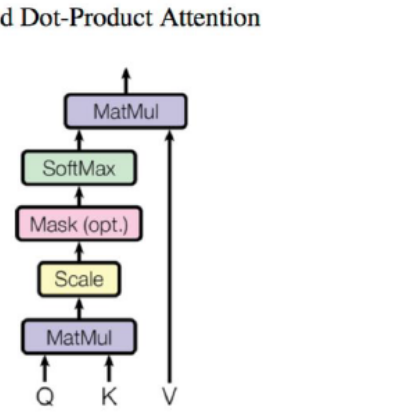
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Self-Attention: Running Example

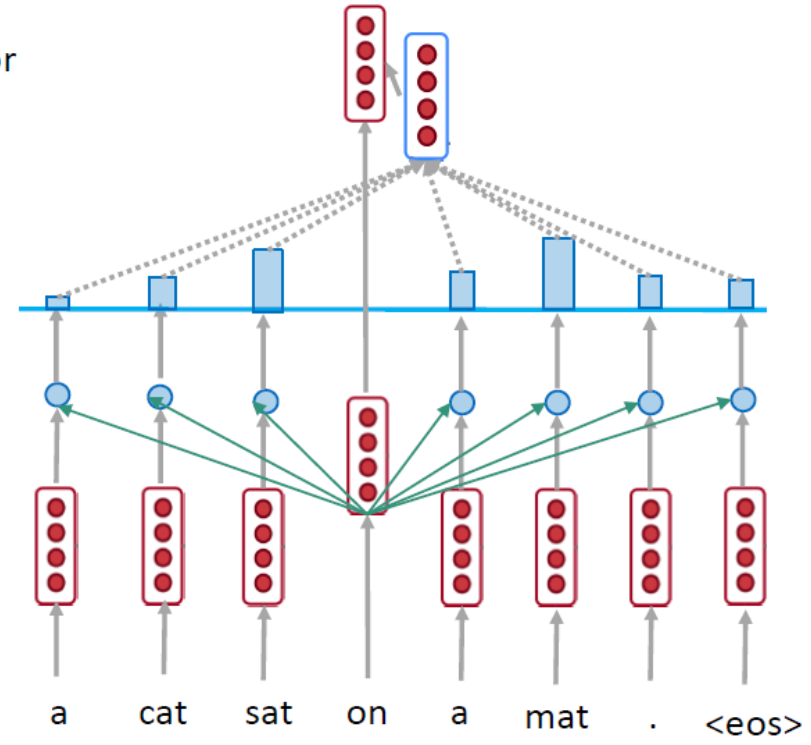


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



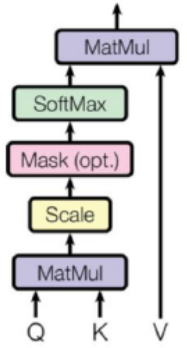
Self-Attention: Running Example

update representation for the word "on"



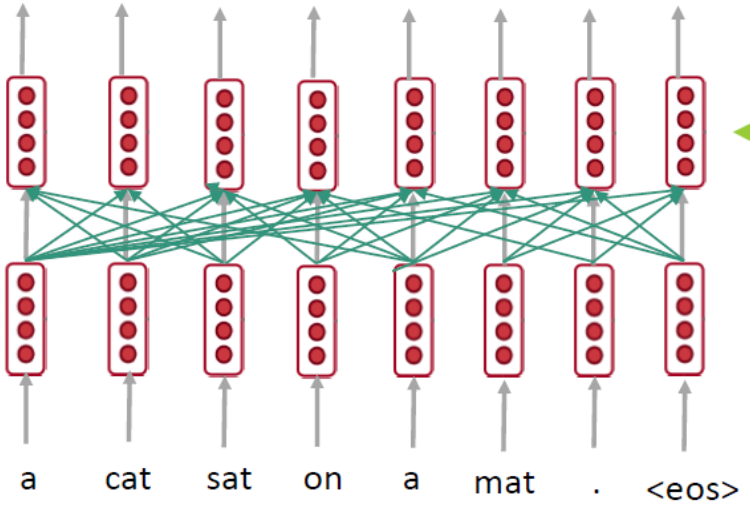
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Scaled Dot-Product Attention

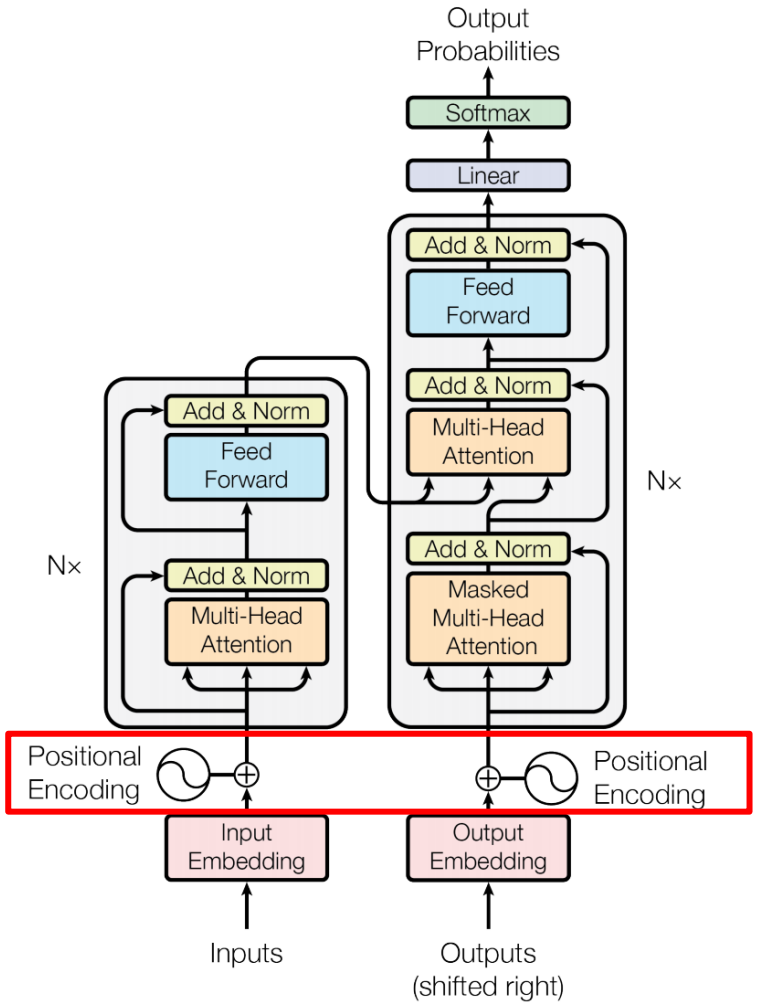


Self-Attention: Running Example

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



Updated representations
for each word
(one layer)



Positional Embedding

Positional Encoding

King

Queen

King = [0.37, 0.99, 0.01, 0.08]

Queen = [0.1, 0.3, 0.2, 0.4]

$$\begin{bmatrix}0.37\\0.99\\0.01\\0.08\end{bmatrix} + \text{Positional Encoding} \rightarrow \begin{bmatrix}0.42\\0.84\\0.12\\0.81\end{bmatrix}$$

Positional Encoding

Positional Encoding

Positional Encoding

Requirements:

1. It should output a unique encoding for each time-step (word's position in a sentence)
2. Distance between any two time-steps should be consistent across sentences with different lengths.
3. Our model should generalize to longer sentences without any efforts. Its values should be bounded.
4. It must be deterministic: Every position should have its own identifier independent of the sequence.
5. Positional encodings should not be too large (otherwise, we may lose semantics)

Positional Encoding

0:	0	0	0	0
1:	0	0	0	1
2:	0	0	1	0
3:	0	0	1	1
4:	0	1	0	0
5:	0	1	0	1
6:	0	1	1	0
7:	0	1	1	1
8:	1	0	0	0
9:	1	0	0	1
10:	1	0	1	0
11:	1	0	1	1
12:	1	1	0	0
13:	1	1	0	1
14:	1	1	1	0
15:	1	1	1	1

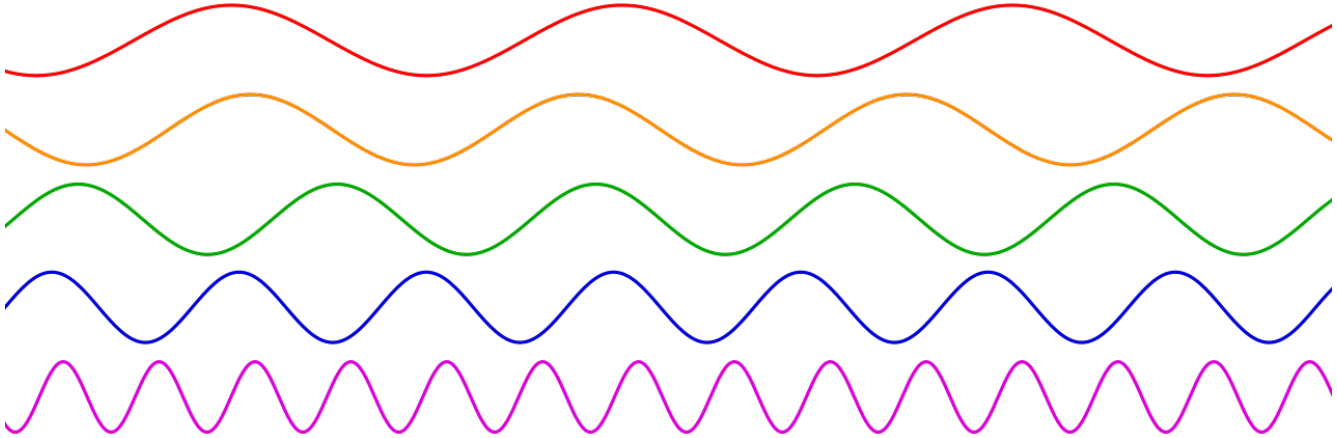
You can spot the rate of change between different bits.

The LSB bit is alternating on every number, the second-lowest bit is rotating on every two numbers, and so on. But using binary values would be a waste of space in the world of floats

Positional Encoding

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

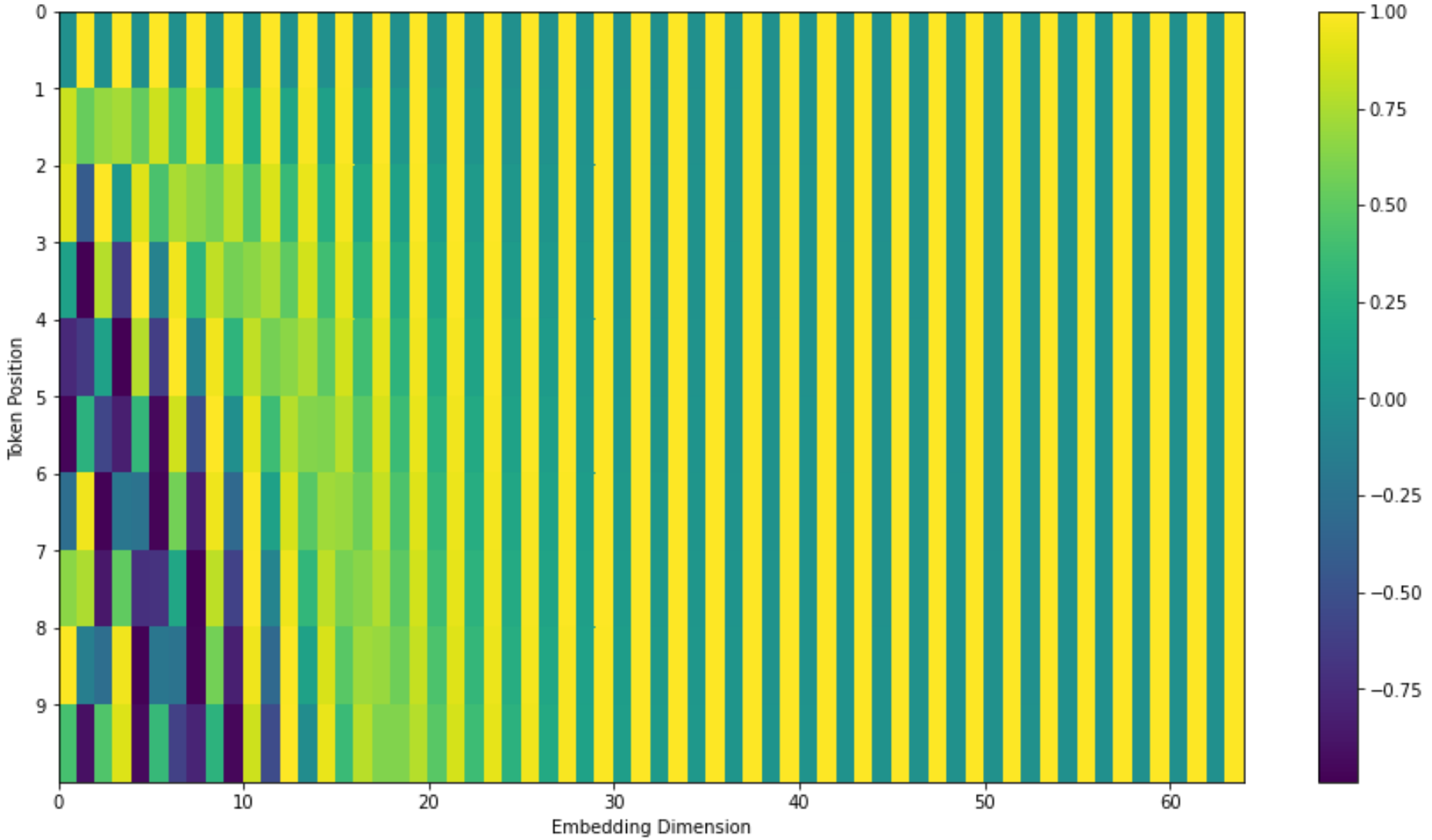


d = 5

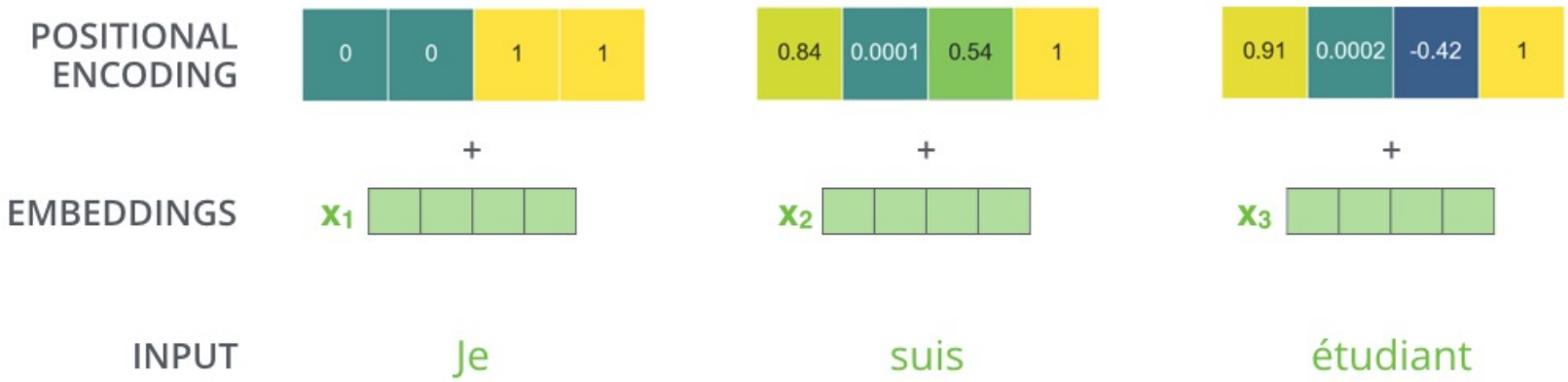
0.42	0.31	0.21	0.31	0.45

Pos = 0 , 1, ...
i = 0, 1, 2, 3, 4

Positional Encoding



Positional Encoding



Summary

What is English?

What is Language?

What is context?

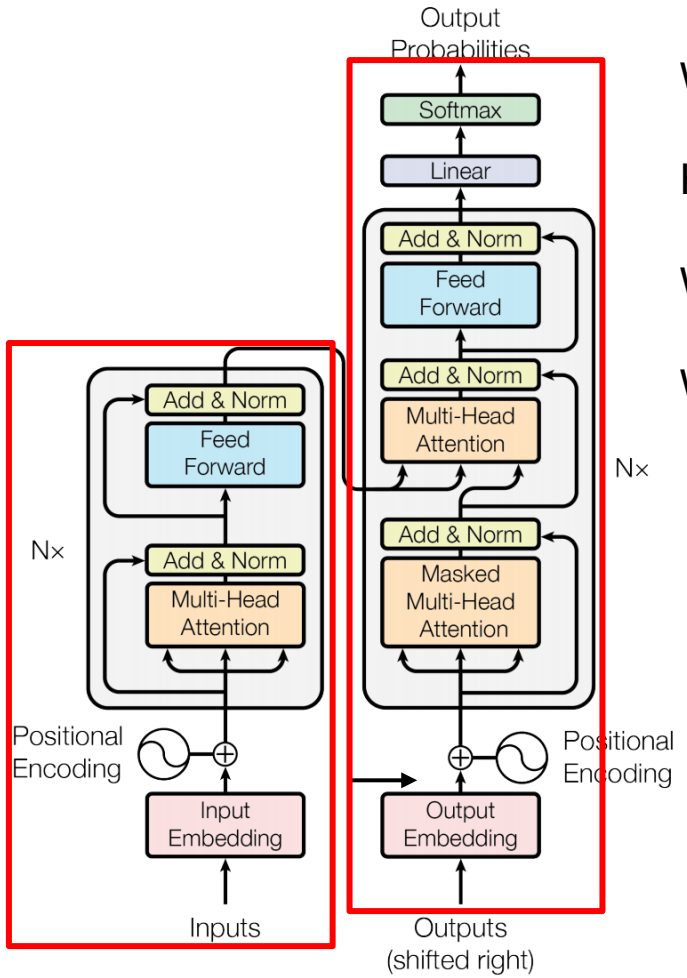


Figure 1: The Transformer - model architecture.

What is French?

How to map English to French?

What is Language?

What is context?

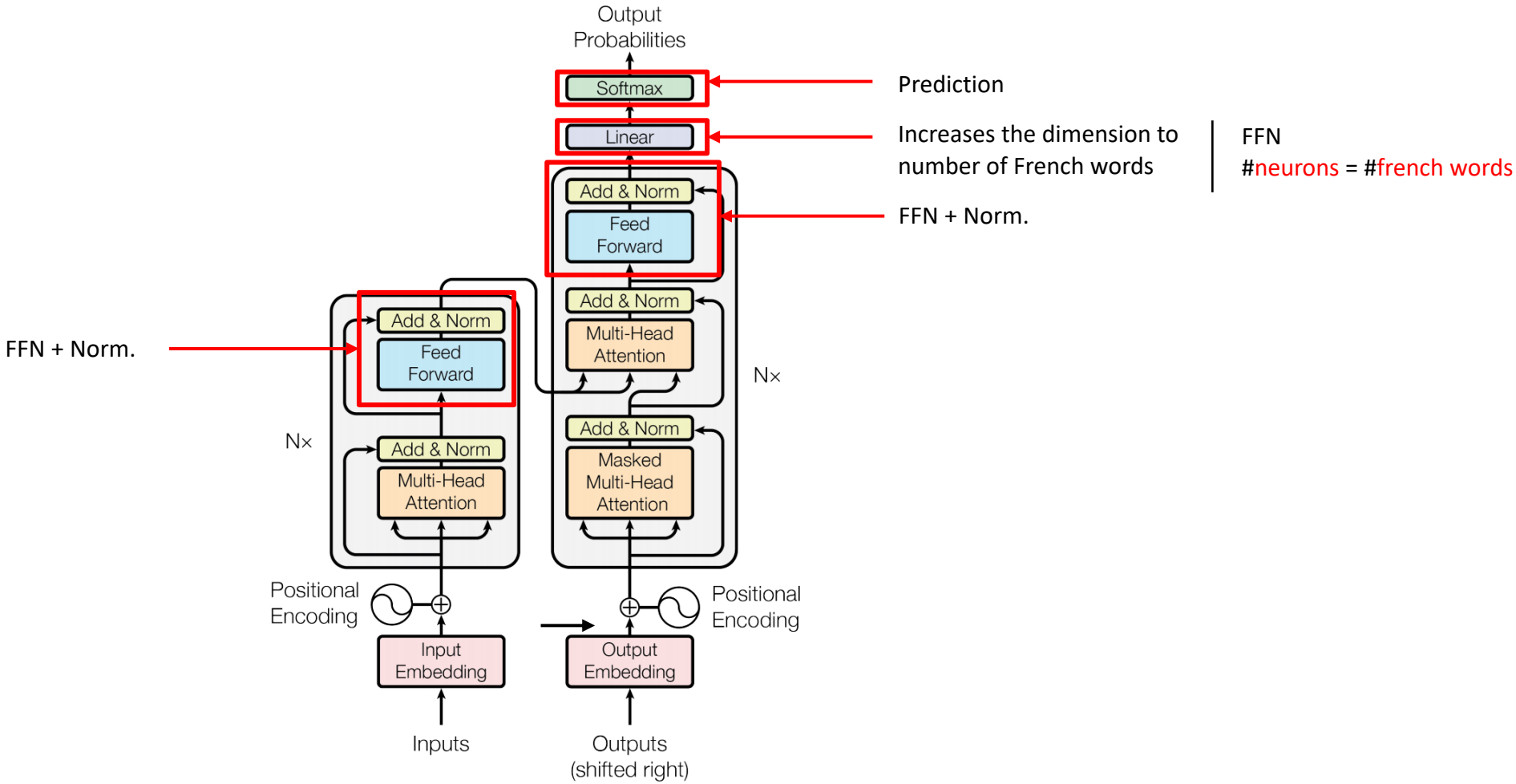


Figure 1: The Transformer - model architecture.

Comparison

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$

Comparison

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

BLEU Score

Proceedings of the 40th Annual Meeting of the Association for
Computational Linguistics (ACL), Philadelphia, July 2002, pp. 311-318.

BLEU: a Method for Automatic Evaluation of Machine Translation

Kishore Papineni, Salim Roukos, Todd Ward, and **Wei-Jing Zhu**
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598, USA
{papineni,roukos,toddward,weijing}@us.ibm.com

Abstract

Human evaluations of machine translation are extensive but expensive. Human evaluations can take months to finish and involve human labor that can not be reused. We propose a method of automatic machine translation evaluation that is quick, inexpensive, and language-independent, that correlates highly with human evaluation, and that has little marginal cost per run. We present this method as an automated understudy to skilled human judges which substitutes for them when there is need for quick or frequent evaluations.[1]

the evaluation bottleneck. Developers would benefit from an inexpensive automatic evaluation that is quick, language-independent, and correlates highly with human evaluation. We propose such an evaluation method in this paper.

1.2 Viewpoint

How does one measure translation performance? *The closer a machine translation is to a professional human translation, the better it is.* This is the central idea behind our proposal. To judge the quality of a machine translation, one measures its closeness to one or more reference human translations according to a numerical metric. Thus, our MT evaluation system requires two ingredients:

BLEU Score

[Input] French: Le chat est sur le tapis

Reference 1: The cat is on the mat
Reference 2: There is a on the mat

MT Output 1: The The The The The The

$$ModifiedPrecision\ \frac{\#Overlaps_{Clipped}}{\#GeneratedWords}$$

BLEU Score

[Input] French: Le chat est sur le tapis

Reference 1: The cat is on the mat
Reference 2: There is a on the mat

MT Output 1: The The The The The The

$$ModifiedPrecision\ \frac{\#Overlaps_{Clipped}}{\#GeneratedWords} = \frac{\sum_{Unigram} Count_{clipped}}{\sum_{Unigrams} Count}$$

BLEU Score on Bigrams

[Input] French: Le chat est sur le tapis

Reference 1: The cat is on the mat
Reference 2: There is a on the mat

MT Output 1: The cat the cat on the mat

	$\#Count$	$\#Count_{clipped}$
The cat	2	1
cat the	1	0
Cat on	1	1
On the	1	1
The mat	1	1

$$ModifiedPrecision\ \frac{\#Overlaps_{Clipped}}{\#Bigrams} = \frac{\sum_{Bigrams} Count_{clipped}}{\sum_{Bigrams} Count}$$

BLEU Score: Summary

$$P_n = \frac{\sum_{N-gram} Count_{clipped}}{\sum_{N-gram} Count}$$

$$P_1 = \frac{\sum_{1-gram} Count_{clipped}}{\sum_{1-gram} Count}$$

$$P_2 = \frac{\sum_{2-gram} Count_{clipped}}{\sum_{3-gram} Count}$$

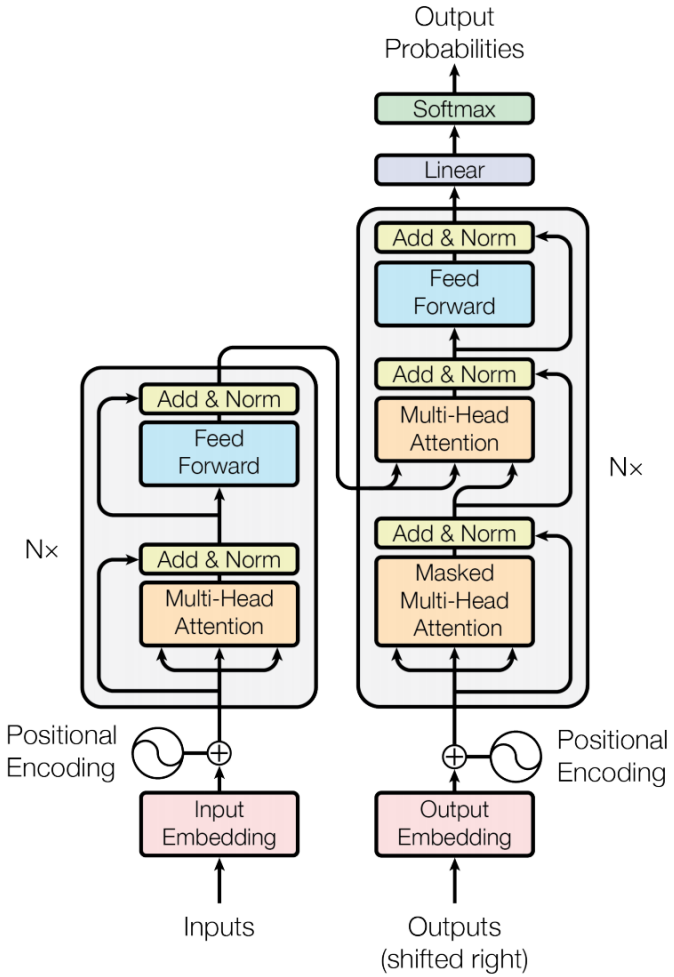
$$P_3 = \frac{\sum_{3-gram} Count_{clipped}}{\sum_{3-gram} Count}$$

$$P_4 = \frac{\sum_{4-gram} Count_{clipped}}{\sum_{4-gram} Count}$$

1. Take the average over different P_1
2. Use exp to
3. Add a penalty for short outputs

$$P_1 = P * e^{(P_1+P_2+P_3+P_4)/4}$$

Transformer Architecture



Is there a recurrence?

- **Training:**

  Teacher forcing is a procedure in which during training the model receives the ground truth output y(t) as input at time t + 1.

- This can help us parallelize the training
- It's fine to have depends on the previous output in the testing time