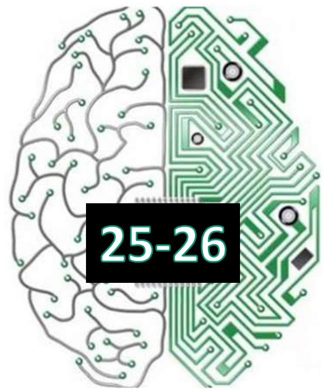
Open Elective Course [OE]
Course Code: CSO507
Winter 2023-24

Lecture#

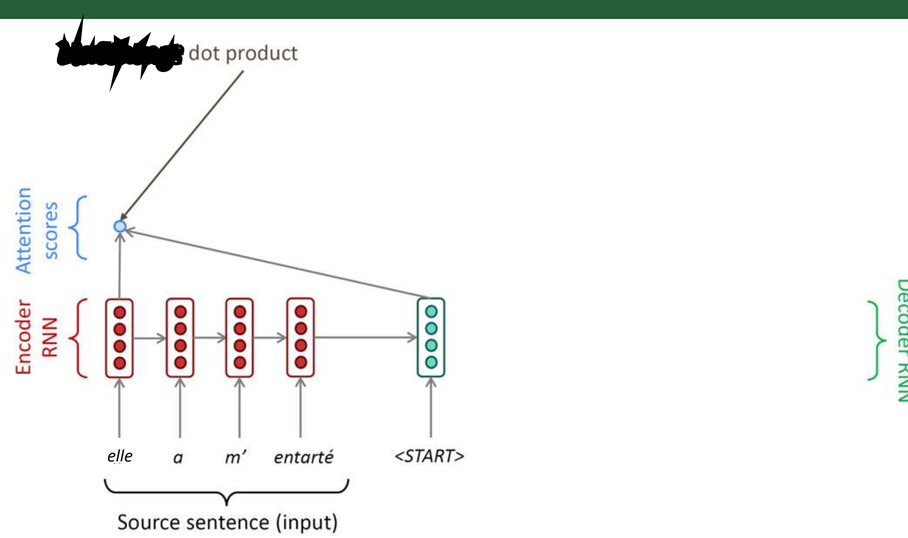


Deep Learning

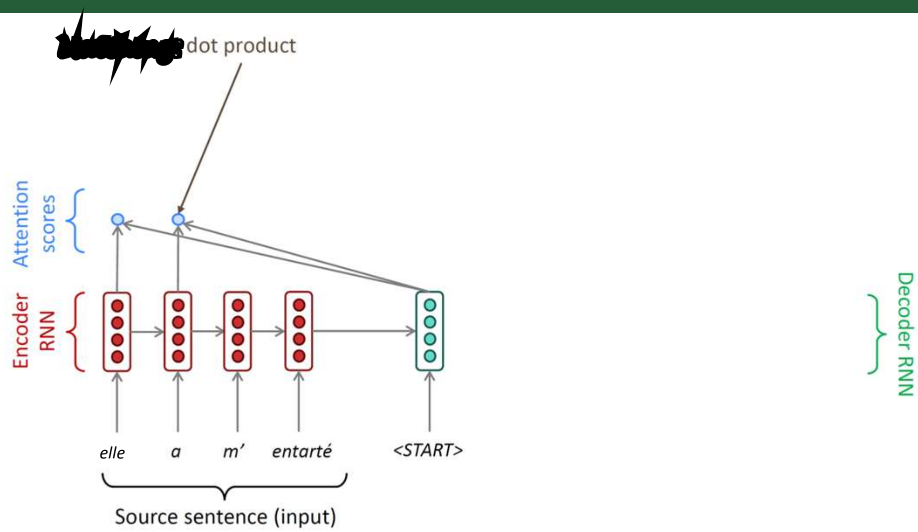
Unit-5: Sequence Modeling with Recurrent Neural Network (RNN) Part-VI&VII, Attention Mechanism, Transformer (Part-I)

Course Instructor:
Dr. Monidipa Das
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology (Indian School of Mines) Dhanbad, Jharkhand 826004, India

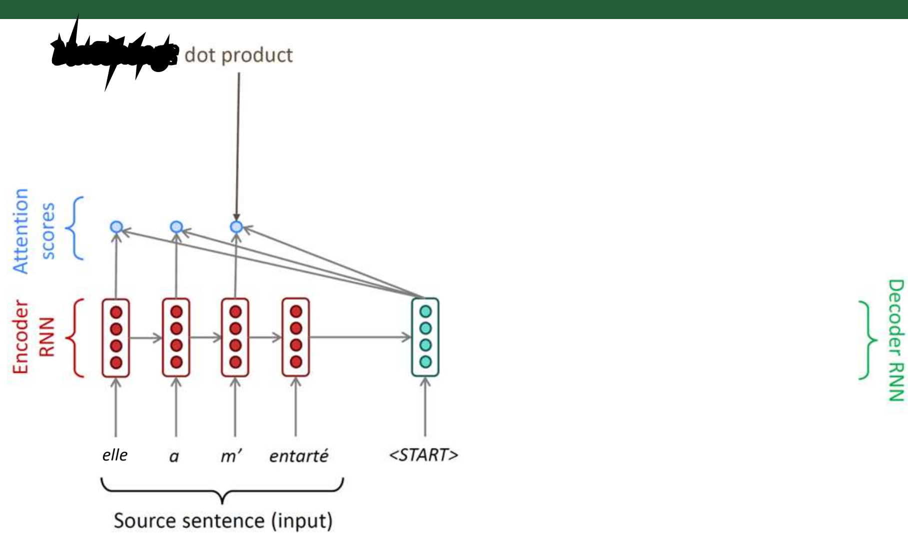
Attention Mechanism



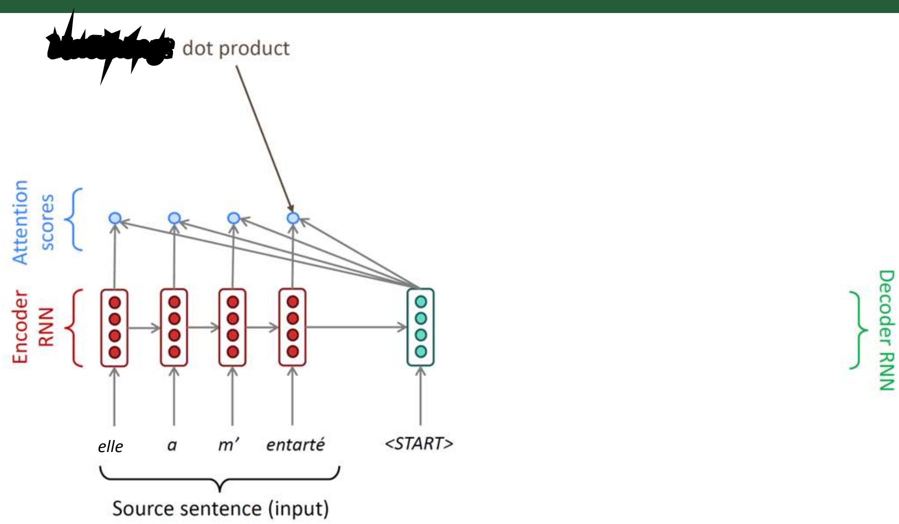
Attention Mechanism



Attention Mechanism

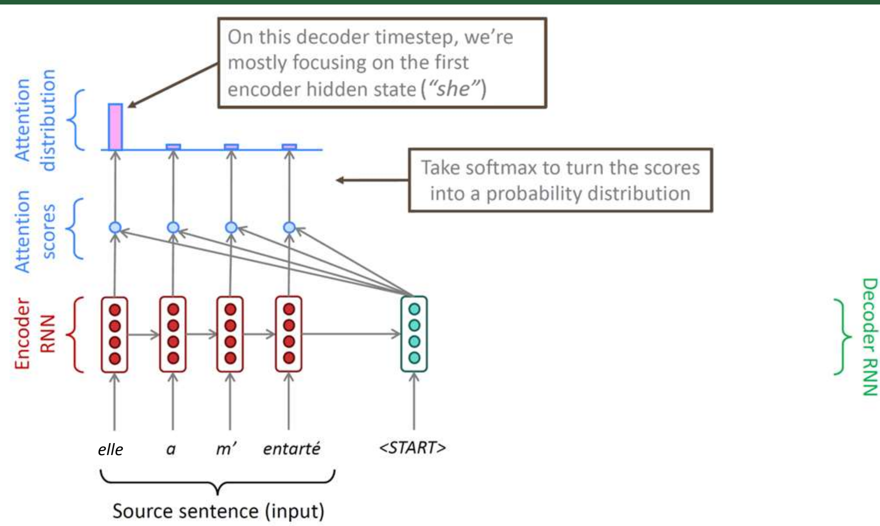


Attention Mechanism



Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

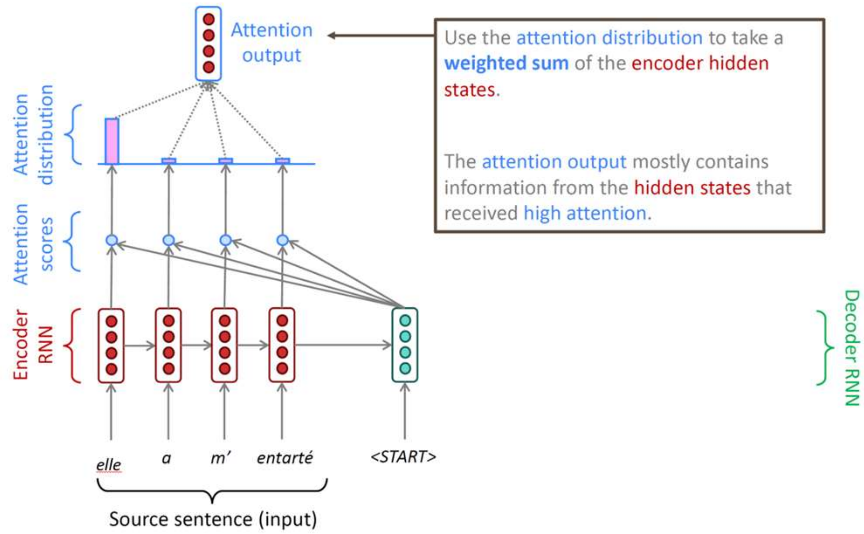
Attention Mechanism



Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Attention Mechanism

7



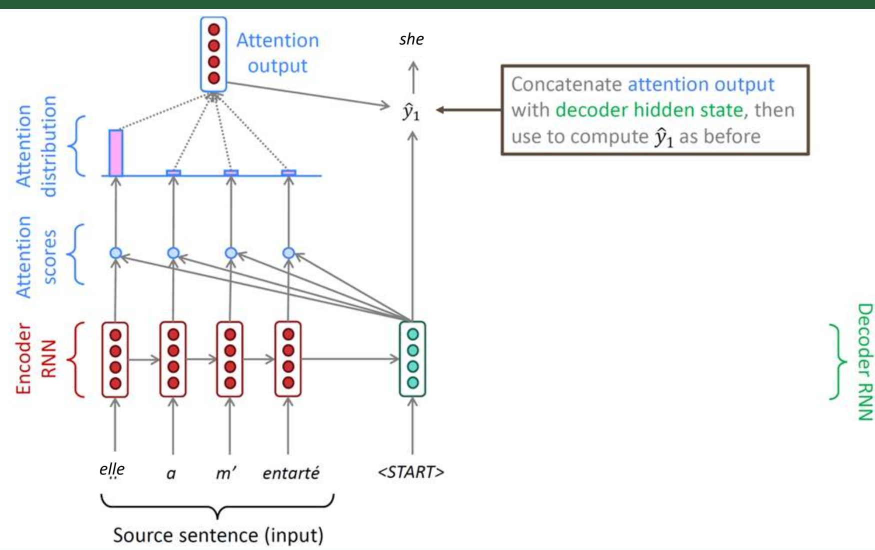
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information from the hidden states that received high attention.

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Attention Mechanism

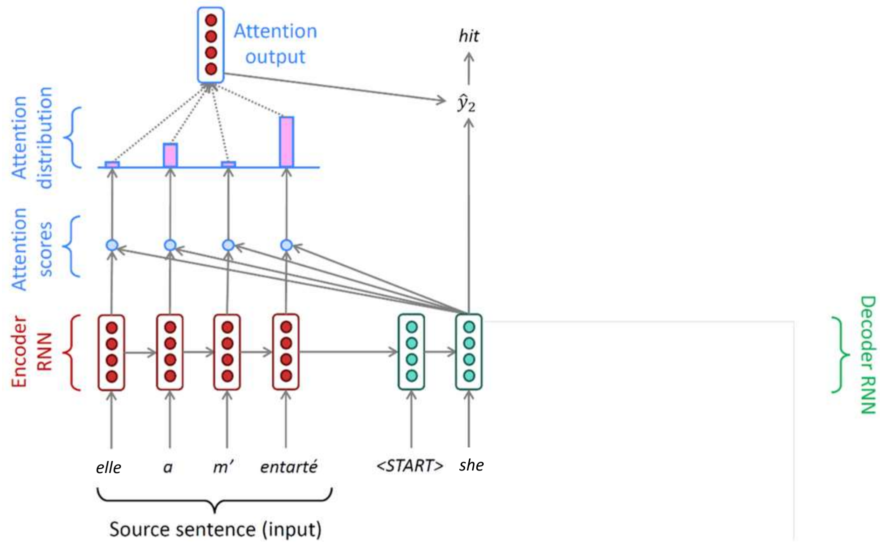
8



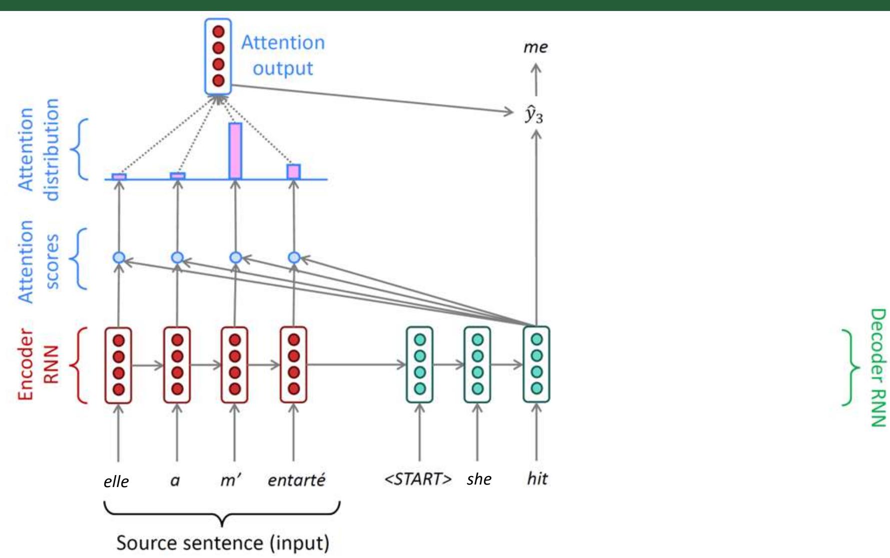
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Attention Mechanism

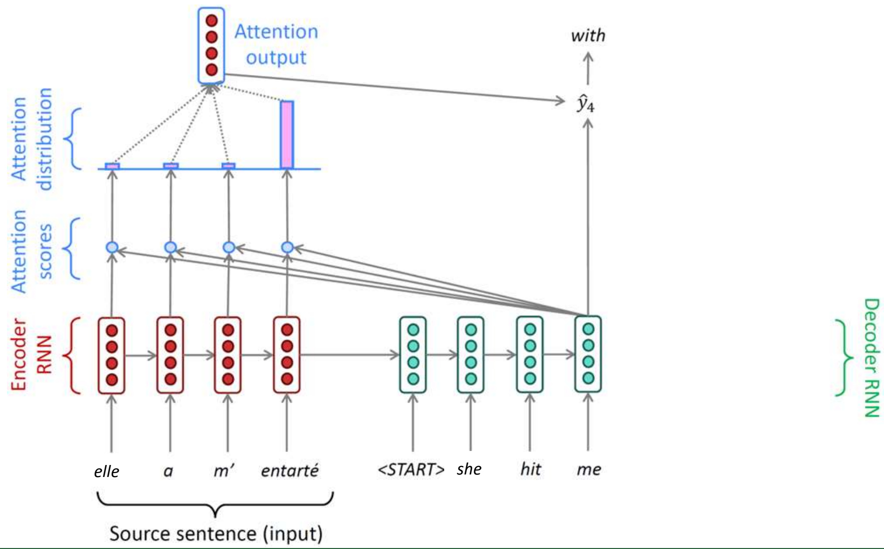


Attention Mechanism



Attention Mechanism

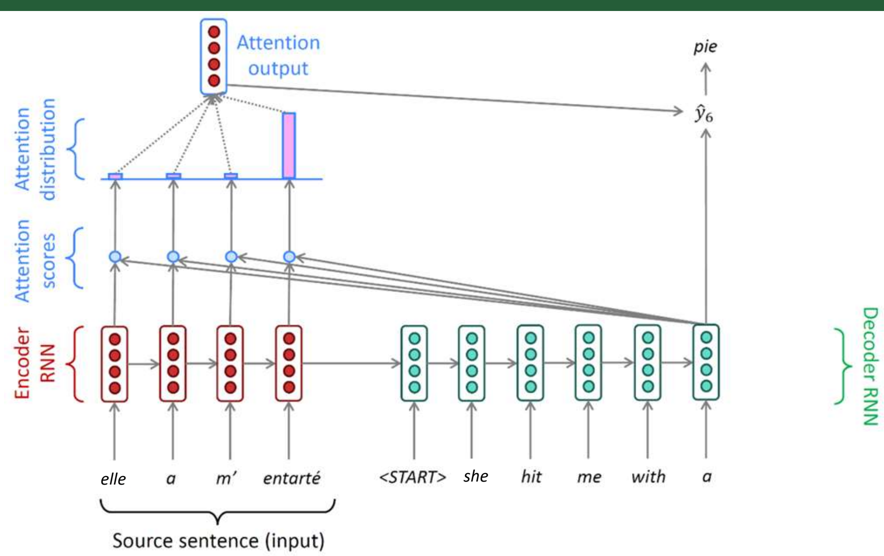
11



Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Attention Mechanism

12



Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Attention Mechanism

13

- Input sequence X, encoder f_{enc}, and decoder f_{dec}
- $f_{enc}\,(X)$ produces hidden states $h_1^{enc}, h_2^{enc}, \ldots, h_N^{enc}$
- On time step t, we have decoder hidden state h_t
- Compute attention score $\alpha_i = h_t^T h_i^{enc}$ //dot product attention
- Compute attention distribution $\hat{\alpha}_i = P_{att}(X_i) = softmax(\alpha_i)$
- Attention output: $h_{att}^{enc} = \sum_i \hat{\alpha}_i h_i^{enc}$
- $Y_t \sim g(h_t, h_{att}^{enc}; \theta)$
  - Sample an output using both

Attention Mechanism

14

- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on

Self Attention

15

“I am going to the bank to deposit the cheque”
“The boat went down the river bank”

The vector representation used as input to RNN cannot differentiate context

Problems with RNNs: **Sequential computation inhibits parallelization**

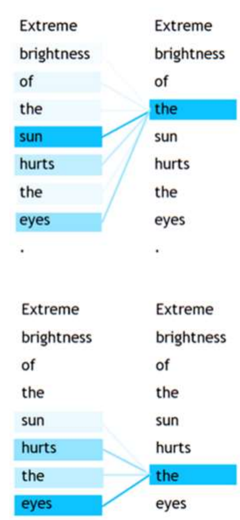
Self Attention

16

- **Self attention**
  - Re-express representation in terms of the context the word occurs in

  - Construct Probability distribution of importance - Dot product and normalization with only the input

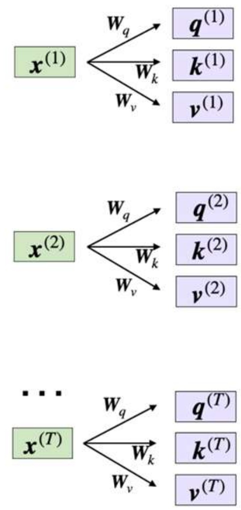
  - Re-express word vectors weighted by the probabilities as weights



Self Attention

17

- **Defining the Weight Matrices**
  - Three matrices serve to project the inputs into ***query, key***, and ***value*** components

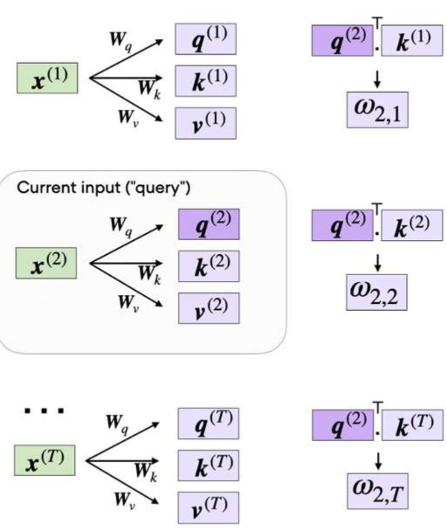


- Query sequence: $\mathbf{q}^{(i)} = \mathbf{W}_q \mathbf{x}^{(i)}$ for $i \in [1, T]$
- Key sequence: $\mathbf{k}^{(i)} = \mathbf{W}_k \mathbf{x}^{(i)}$ for $i \in [1, T]$
- Value sequence: $\mathbf{v}^{(i)} = \mathbf{W}_v \mathbf{x}^{(i)}$ for $i \in [1, T]$

Self Attention

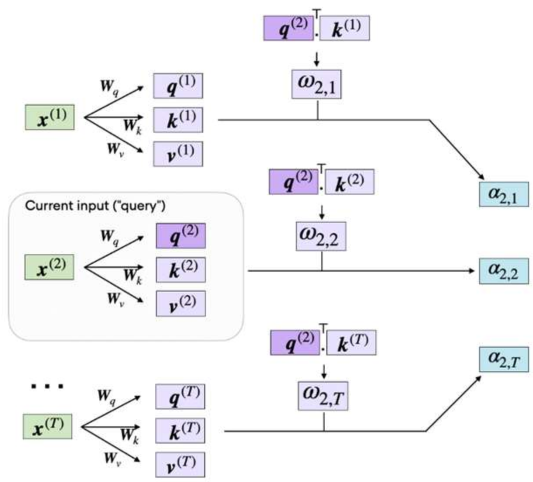
18

- **Computing the Unnormalized Attention Weights**



Self Attention

- **Computing the Attention Scores**

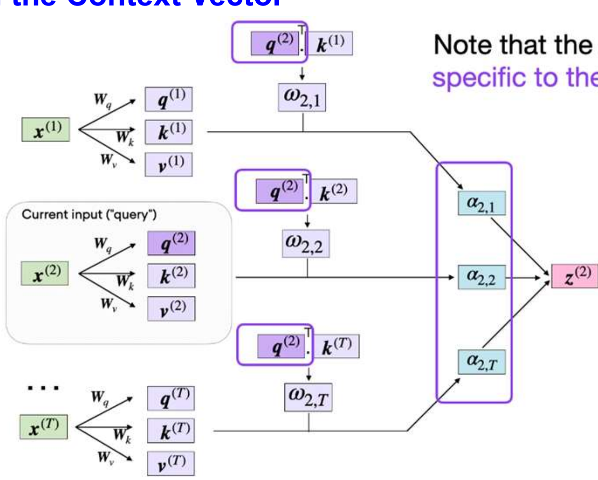


where $\alpha_{2,i} = \text{softmax}\left(\frac{\omega_{2,i}}{\sqrt{d_k}}\right)$

Self Attention

- **Computing the Context Vector**

Note that the attention scores are specific to the current input token



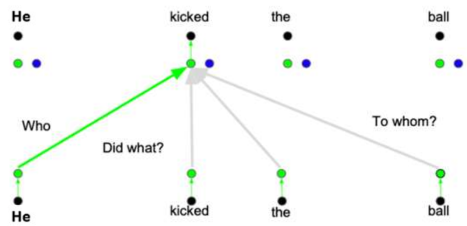
where $z^{(2)} = \sum_{j=1}^{T} \boxed{\alpha_{2,j}}\, v^{(j)}$

Multi-Head Attention

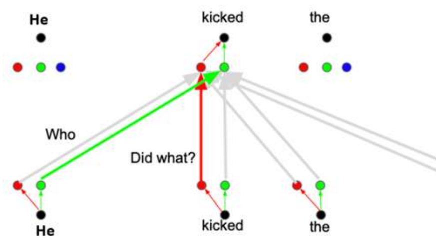
21

- All similarities do not encode the same aspects
- Multi-head Attention: Each notion of similarity is represented by an attention head

"He kicked the ball"



Aspect: "who"



Aspect: "did what"

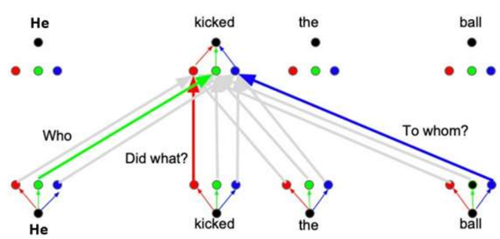
Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Multi-Head Attention

22

- All similarities do not encode the same aspects
- Multi-head Attention: Each notion of similarity is represented by an attention head

"He kicked the ball"

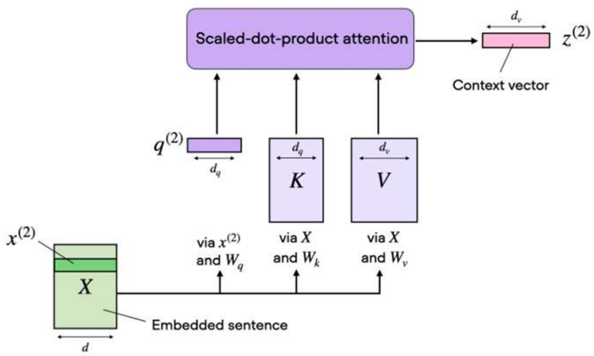


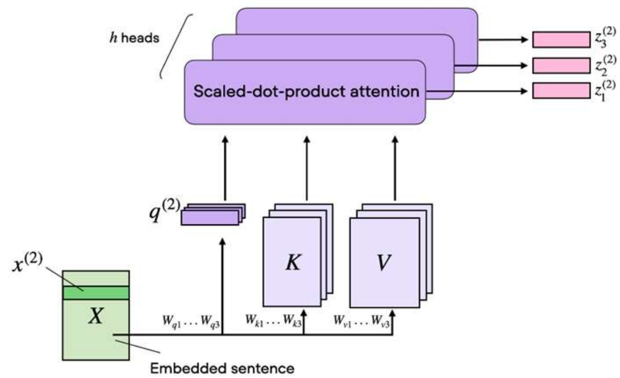
Aspect: "to whom"

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Multi-Head Attention

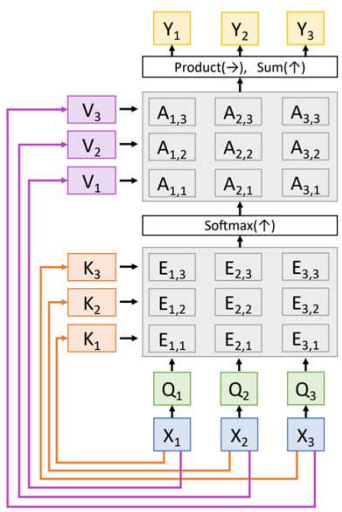
23



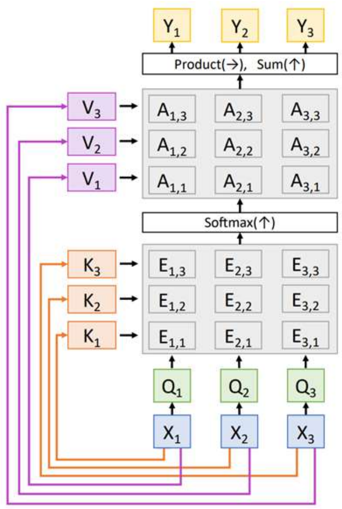


Self Attention: Permutation Invariant!

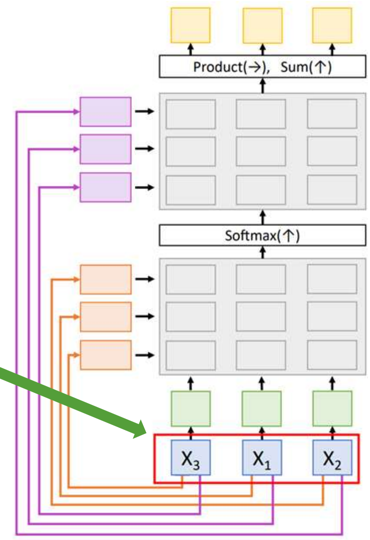
24



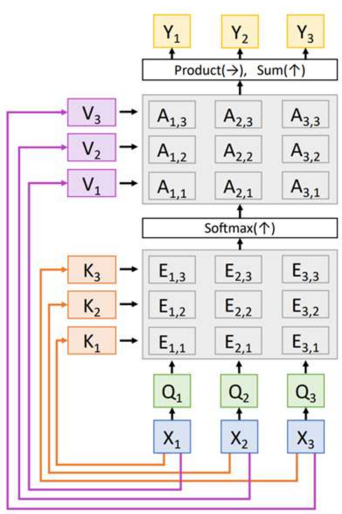
Self Attention: Permutation Invariant!



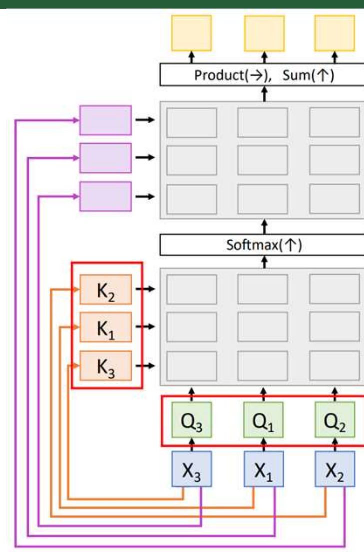
Consider permutating the input vectors:



Self Attention: Permutation Invariant!

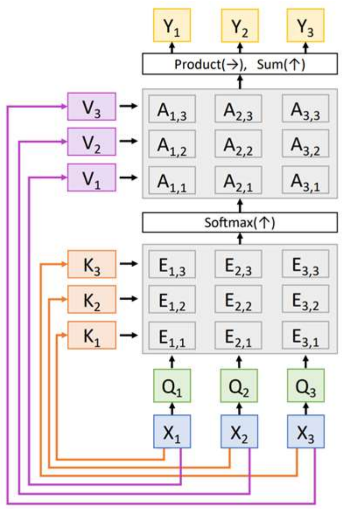


Consider permutating the input vectors:

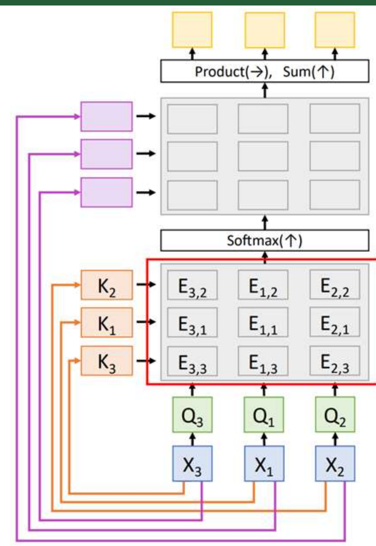


Self Attention: Permutation Invariant!

27

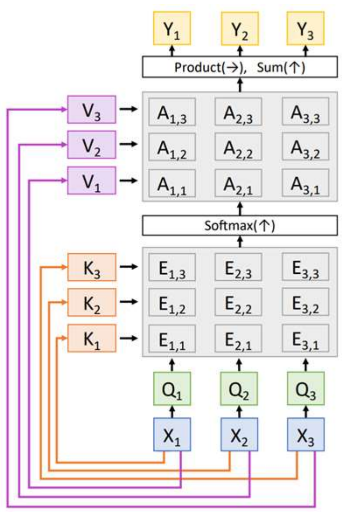


Consider permutating the input vectors:

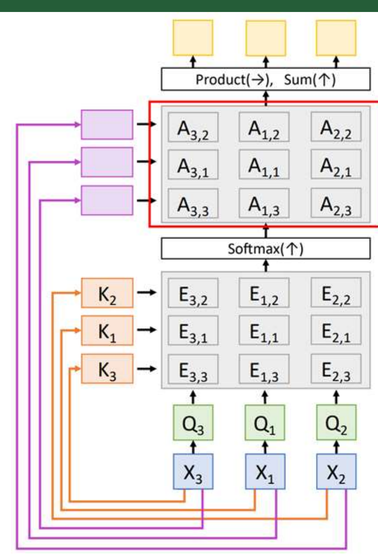


Self Attention: Permutation Invariant!

28

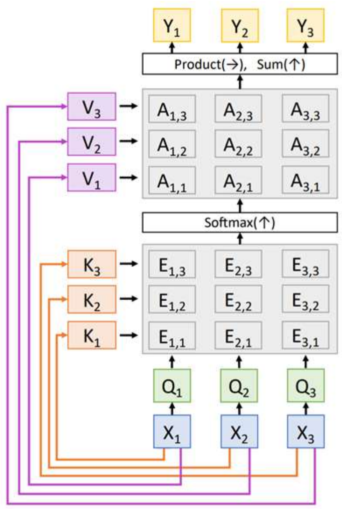


Consider permutating the input vectors:

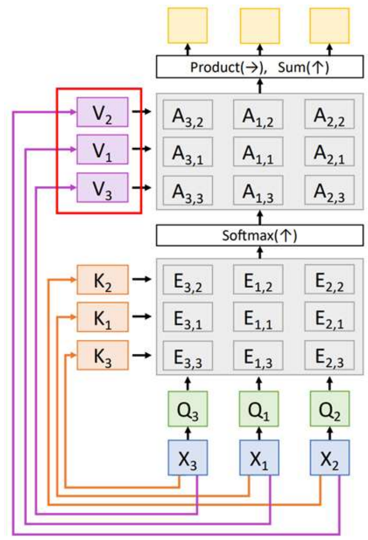


Self Attention: Permutation Invariant!

29

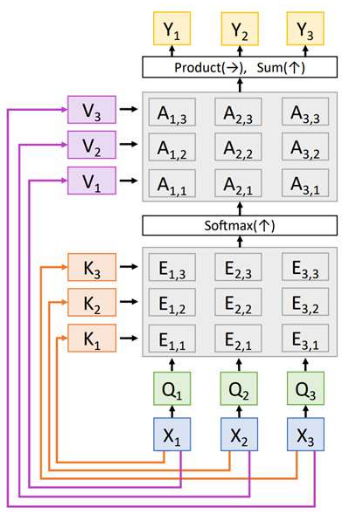


Consider permutating the input vectors:



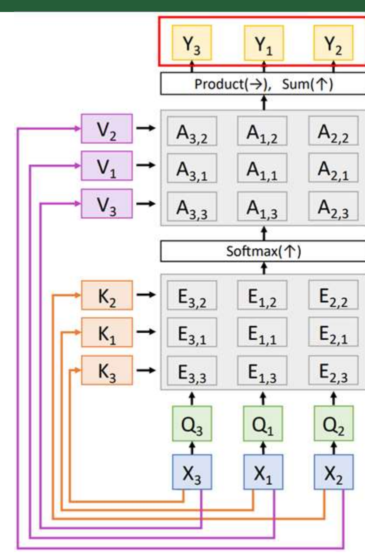
Self Attention: Permutation Invariant!

30



Consider permutating the input vectors:

The output is the same put permuted!



Self Attention: Permutation Invariant!

31

- Self attention doesn't "know" the order of the vectors it is processing!
- But what if the ordering of the input vectors conveys information as well?
  - The position of a word in a sentence matters!

    "The man ate a fish" **"The fish ate a man""**

- Solution: **Positional Encoding**

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

Self Attention: Permutation Invariant!

32

- Add positional embeddings to input embeddings
  - Same dimension
  - Can be learned or fixed

Options for *pos*(.)

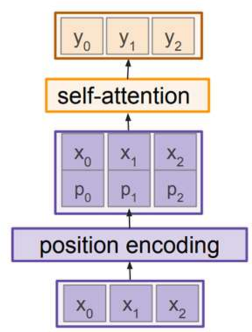
1. Learn a lookup table:
   - Learn parameters to use for *pos*(t) for t ε [0, T)
   - Lookup table contains T x d parameters.
2. Design a fixed function with the desiderata

Desiderata of *pos*(.) :

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**.

$$\mathbf{pos(j)} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_d$$

where $\omega_k = \frac{1}{10000^{2k/d}}$



Concatenate special positional encoding p_j to each input vector x_j

We use a function *pos*: $N \rightarrow R^d$ to process the position j of the vector into a d-dimensional vector

So, p_j = *pos*(j)

Prof. Monidipa Das, Department of CSE, IIT (ISM) Dhanbad

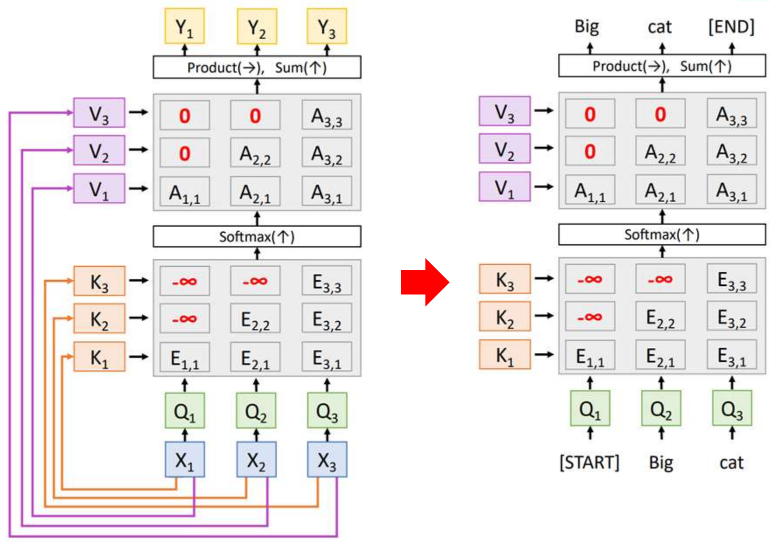
Masked Self-Attention Layer

33

Prevent vectors from looking at future vectors.

Manually set alignment scores to –infinity (−∞)

Used for language modeling (predict next word)

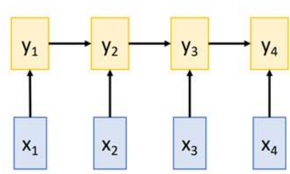


RNN vs. 1D Convolution vs. Self-Attention

34

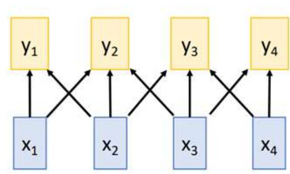
- **Three Ways of Processing Sequences**

Recurrent Neural Network



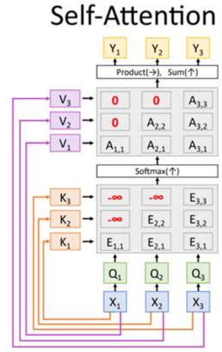
Works on **Ordered Sequences**
(+) Good at long sequences: After one RNN layer, h_T "sees" the whole sequence
(-) Not parallelizable: need to compute hidden states sequentially

1D Convolution



Works on **Multidimensional Grids**
(-) Bad at long sequences: Need to stack many conv layers for outputs to "see" the whole sequence
(+) Highly parallel: Each output can be computed in parallel

Self-Attention



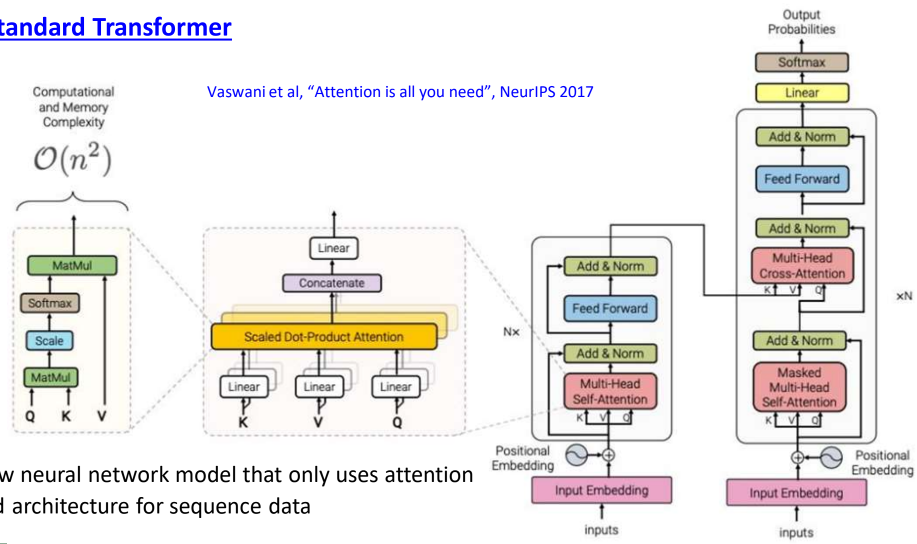
Works on **Sets of Vectors**
(-) Good at long sequences: after one self-attention layer, each output "sees" all inputs!
(+) Highly parallel: Each output can be computed in parallel
(-) Very memory intensive

The Transformer

35

- **[Architecture of a Standard Transformer]**

Vaswani et al, "Attention is all you need", NeurIPS 2017



- Transformers are a new neural network model that only uses attention
- a fully attention-based architecture for sequence data

Transformer Block

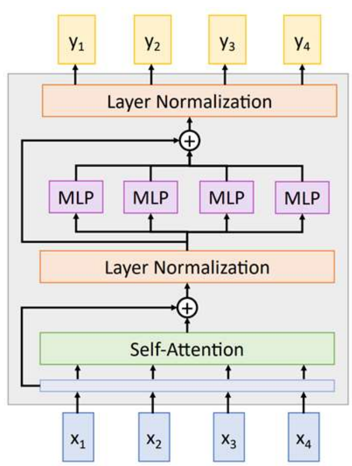
36

Transformer Block:
Input: Set of vectors x
Output: Set of vectors y

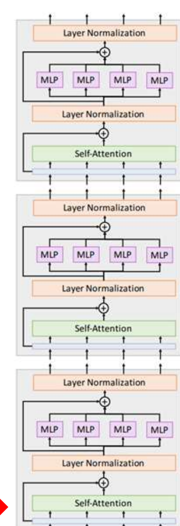
Self-attention is the only interaction between vectors!

Layer norm and MLP work independently per vector

Highly scalable, highly parallelizable

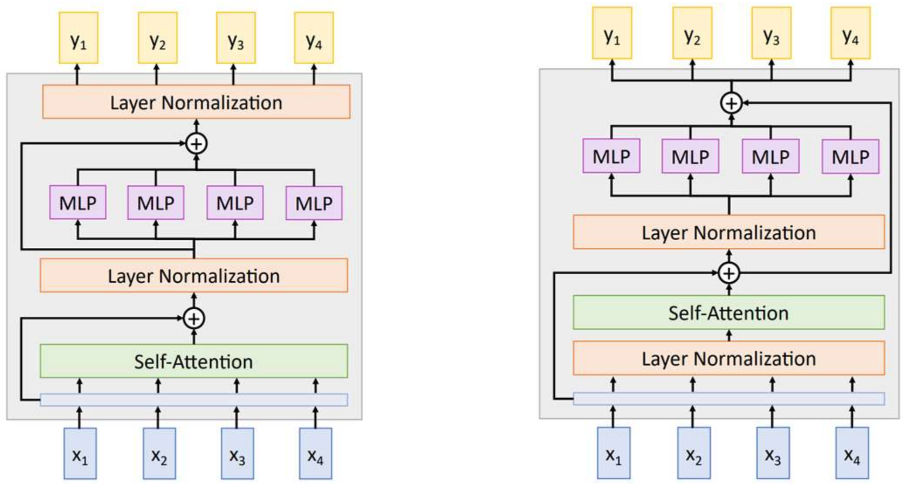


A Transformer is a sequence of transformer blocks



Post-Norm Transformer and Pre-Norm Transformer

37



Baevski & Auli, "Adaptive Input Representations for Neural Language Modeling", arXiv 2018

Scaling up Transformers

38

Model	Layers	Width	Heads	Params
Transformer-Base	12	512	8	65M
Transformer-Large	12	1024	16	213M
BERT-Base	12	768	12	110M
BERT-Large	24	1024	16	340M
XLNet-Large	24	1024	16	~340M
RoBERTa	24	1024	16	355M
GPT-2	48	1600	?	1.5B
Megatron-LM	72	3072	32	8.3B
Turing-NLG	78	4256	28	17B
GPT-3	96	12,288	96	175B
Gopher	80	16,384	128	280B

Data 10.55 TB | 4096x TPUv3 (38 days) **Training time**

Advantages of Transformers

39

- **Great for modelling context**
  - Each token can have access to all other tokens in the sequence
- **A generic architecture:**
  - Operates on any inputs that can be tokenized!
- **Parallelizable**
- **Empirically shown to perform excellently at scale**

Weaknesses of Transformers

40

- **Quadratic complexity**
  - Each token attends to every other token
  - N tokens → N2 operations
  - Prohibitive as the number of tokens increases!
- **Most powerful language models are extremely expensive**
- **Large body of work on more efficient transformers.**
- **Transformers can overfit easily on smaller datasets**

RNNs to Transformer

41

- **RNNs**
  - **LSTMs work reasonably well for long sequences.**
  - **Expects an ordered sequences of inputs**
  - **Sequential computation: subsequent hidden states can only be computed after the previous ones are done.**
- **Transformer:**
  - **Good at long sequences. Each attention calculation looks at all inputs.**
  - **Can operate over unordered sets or ordered sequences with positional encodings.**
  - **Parallel computation: All alignment and attention scores for all inputs can be done in parallel**
  - **Requires a lot of memory: N x M alignment and attention scalers need to be calculated and stored for a single self-attention head.**

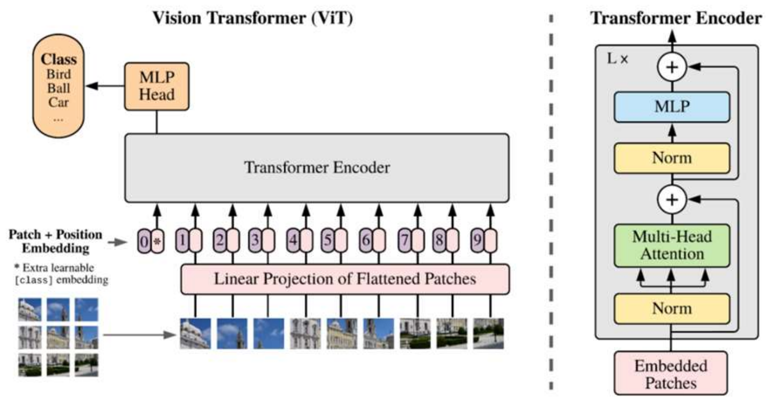
42

Transformers for Vision

Image Classification

43

- **Vision Transformer ('21) [ViT]**
  - Decompose an image to NxN patches and then apply transformer encoder



44

Questions?