# Transformers & Attention

ML Instruction Team, Fall 2022

CE Department
Sharif University of Technology

# Overview

- There has been an array of Transformer based architectures such as BERT, SpanBERT, Transformer-XL, XLNet, GPT-2, etc getting released frequently for the past couple of years.
- The OpenAI's GPT-3 had taken the internet by storm with its ability to perform extremely well on tasks such as QA, Comprehension, even Programming
- But all of this started with a research paper released back in 2017 "Attention is all you need".

# What is a Transformer

- They take a text sequence as input and produce another text sequence as output. eg. to translate an input English sentence to Spanish.
- At its core, it contains a stack of Encoder layers and Decoder layers.

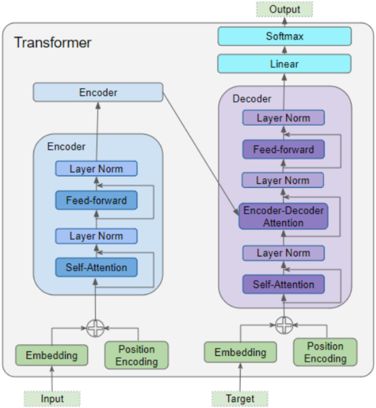



Figure: Transformer schematic consists of Encoder and Decoder.

- As we can observe in the above figure:
  - The Encoder contains the all-important Self-attention layer that computes the relationship between different words in the sequence, as well as a Feed-forward layer.
  - The Decoder contains the Self-attention layer and the Feed-forward layer, as well as a second Encoder-Decoder attention layer.
  - Each Encoder and Decoder has its own set of weights.

# What Does Attention Do ?

- The key to the Transformer's ground-breaking performance is its use of Attention.
- While processing a word, Attention enables the model to focus on other words in the input that are closely related to that word.
- eg. 'Ball' is closely related to 'blue' and 'holding'. On the other hand, 'blue' is not related to 'boy'.

The boy is holding a blue ball

- The Transformer architecture uses self-attention by relating every word in the input sequence to every other word. eg. Consider two sentences:
  - The cat drank the milk because **it** was hungry.
  - The cat drank the milk because **it** was sweet.

- When the model processes the word 'it', self-attention gives the model more information about its meaning so that it can associate 'it' with the correct word.

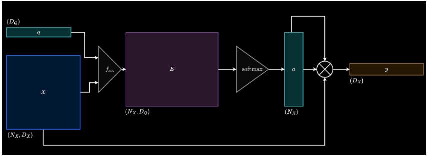

Figure: Implementation of self attention on the example (Dark colors represent higher attention).

- To enable it to handle more nuances about the intent and semantics of the sentence, Transformers include multiple attention scores for each word. For instance:


Figure: Including multiple attention scores for the same example.

# Evolution of Attention

- **Version 0**
  - To understand the intuition of attention, we start with an **input** and a **query**.
  - In terms of computation, **attention is given** to parts of the input matrix which is **similar** to the query vector.
  - $f_{att}$ which is a "feed-forward network". The feed-forward network takes the query and input, and projects both of them to dimension $D_E$.

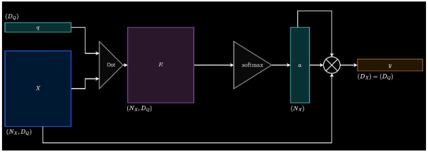



(a) Schematic of attention Version 0

| | Notation | Equation | Shape |
|---|---|---|---|
| Similarity Score | $e$ | $e_i = f_{att}(X_i, q)$ | $(N_X, D_E)$ |
| Attention Weights | $a$ | $a = \text{softmax}(e)$ | $(N_X)$ |
| Output Vector | $y$ | $y = \sum_i a_i X_i$ | $(D_X)$ |

(b) Outputs Table

- **Version 1**
  - The first change we make to the mechanism is swapping out the feed-forward network with a **dot product operation**.
  - Turns out that this is **highly efficient** with reasonably good results.



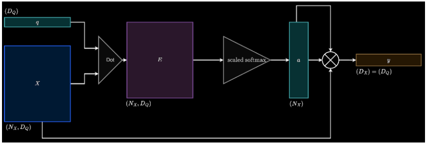

(a) Schematic of attention Version 1

| | Notation | Equation | Shape |
|---|---|---|---|
| Similarity Score | $e$ | $e_i = q.X_i$ | $(N_X, D_Q)$ |
| Attention Weights | $a$ | $a = \text{softmax}(e)$ | $(N_X)$ |
| Output Vector | $y$ | $y = \sum_i a_i X_i$ | $(D_X)$ |

(b) Outputs Table

## Version 2

- This version is a very important concept realized in the original paper. The authors propose **scaled dot product** instead of **normal dot product** as the similarity function.
- This little change can solve many challenges such as **Vanishing Gradient Problem** and **Unnormalized softmax**.



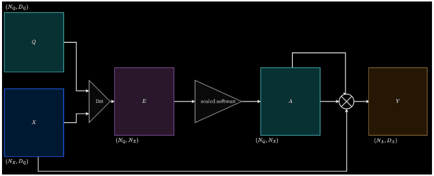

(a) Schematic of attention Version 2

| | Notation | Equation | Shape |
|---|---|---|---|
| Similarity Score | $e$ | $e_i = \frac{q.X_i}{\sqrt{D_Q}}$ | $(N_X, D_Q)$ |
| Attention Weights | $a$ | $a = \mathrm{softmax}(e)$ | $(N_X)$ |
| Output Vector | $y$ | $y = \sum_i a_i X_i$ | $(D_X)$ |

(b) Outputs Table

## Version 3

- Previously we looked at a single query vector. Here we scale this implementation to **multiple query vectors**.
- We calculate the similarities of the input matrix with all the query vectors (query matrix) we have.



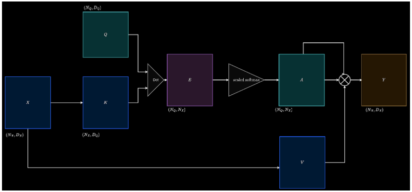

(a) Schematic of attention Version 3

| | Notation | Equation | Shape |
|---|---|---|---|
| Similarity Score | $E$ | $E = \frac{QX^T}{\sqrt{D_Q}}$ | $(N_Q, N_X)$ |
| Attention Weights | $A$ | $A = \mathrm{softmax}(E)$ | $(N_Q, N_X)$ |
| Output Vector | $Y$ | $Y = AX$ | $(N_X, D_X)$ |

(b) Outputs Table

## Version 4 (Cross-Attention)

- To build cross-attention, we make some changes. The changes are specific to the input matrix. As we already know, attention needs an input matrix and a query matrix.
- Suppose we projected the input matrix into a pair of matrices, namely the **key** and **value** matrices.
- This is done to **decouple** the complexity. The input matrix can now have a better projection that takes care of building attention weights and better output matrices as well.

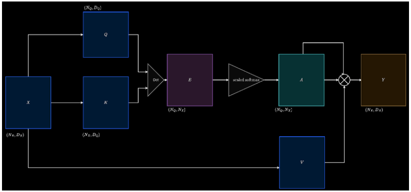



(a) Schematic of attention Version 4

| | Notation | Equation | Shape |
|---|---|---|---|
| Similarity Score | $E$ | $E = \frac{QK^T}{\sqrt{D_Q}}$ | $(N_Q, N_X)$ |
| Attention Weights | $A$ | $A = \mathrm{softmax}(E)$ | $(N_Q, N_X)$ |
| Output Vector | $Y$ | $Y = AX$ | $(N_X, D_X)$ |

(b) Outputs Table

- **Version 5 (Self-Attention)**
  - Like the Version 4 that the key and value matrix are projected versions of the input matrix. What if the query matrix also was projected from the input?
  - Here the main motivation is to build a richer implementation of self with respect to self. This sounds funny, but it is highly important and forms the basis of the Transformer architecture.




(a) Schematic of attention Version 5

| | Notation | Equation | Shape |
|---|---|---|---|
| Similarity Score | $E$ | $E = \frac{QK^T}{\sqrt{D_Q}}$ | $(N_Q, N_X)$ |
| Attention Weights | $A$ | $A = \mathrm{softmax}(E)$ | $(N_Q, N_X)$ |
| Output Vector | $Y$ | $Y = AX$ | $(N_X, D_X)$ |

(b) Outputs Table

# Training the Transformer

- Training data consists of two parts:
  - The source or input sequence (eg. “You are welcome” in English, for a translation problem).
  - The destination or target sequence (eg. “De nada” in Spanish).
- The Transformer’s goal is to learn how to output the target sequence, by using both the input and target sequence.

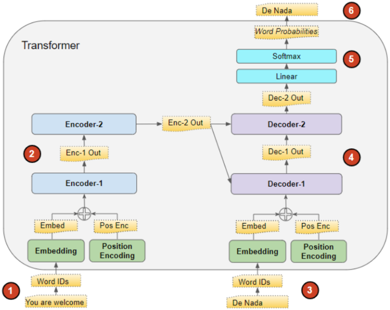



Figure: Training the transformer.