POST 17 - GEN AI

UNDERSTANDING THE MATH BEHIND "SELF ATTENTION MECHANISM"

Generative AI Deep Dives,
Key concepts for Transformers - Part 4

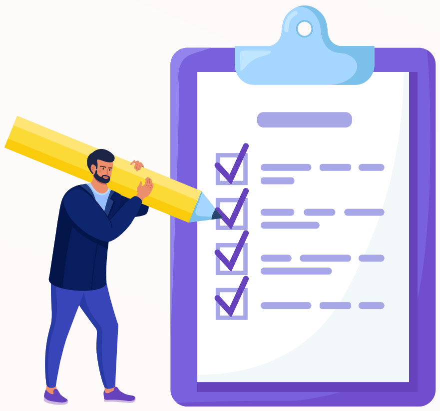
GENERATIVE AI FOR ALL

in DINESHLAL

DINESH LAL
(DIRECTOR, DATA SCIENCE)

WHAT IS COVERED IN THIS DOCUMENT?

- This document explains the topic **Self Attention Mechanism** in detail
- First page covers the flow chart, to understand the overall Mathematical process
- Next section covers step by step explanation of the key steps shown in Flow chart
- In the last section, explanation is shared with example



SELF ATTENTION - FLOW IN ONE PAGE

INPUT WORDS: [WORD1, WORD2, WORD3]

↓

VECTORS: [[VECTOR1], [VECTOR2], [VECTOR3]]

↓

TRANSFORMED BY Q, K, V MATRICES

↓

SCORES: [SCORE1, SCORE2, SCORE3]

↓

SCALED SCORES: [SCALEDSCORE1, SCALEDSCORE2, SCALEDSCORE3]

↓

SOFTMAX: [WEIGHT1, WEIGHT2, WEIGHT3]

↓

WEIGHTED SUM: [NEWVECTOR1, NEWVECTOR2, NEWVECTOR3]

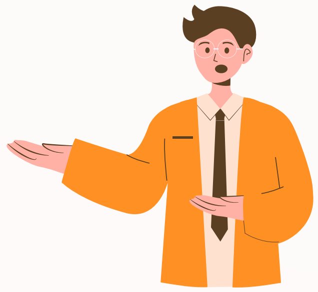
↓

OUTPUT: [OUTPUTVECTOR1, OUTPUTVECTOR2, OUTPUTVECTOR3]

SELF ATTENTION - KEY STEPS EXPLAINED

Vectors and Matrices:

- **Vector:** In the context of NLP, a vector is a numerical representation of a word.*
    - It captures semantic meaning and context.
    - For example, the word "apple" might be represented as [0.2, -0.4].
- **Matrices:** These are grids of numbers (or functions) that transform vectors in certain ways.
    - In self-attention, we have three matrices called Query (Q), Key (K), and Value (V).

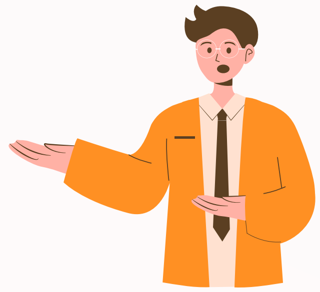


*Word Embeddings Explained in Post 16

SELF ATTENTION - KEY STEPS EXPLAINED

Vectors and Matrices:

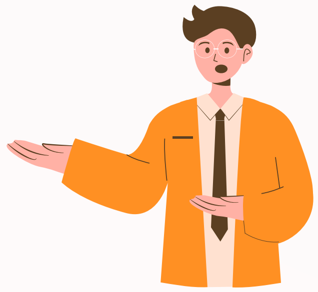
- Each matrix transforms every word vector into a new vector:
  - **Query Vector:** Determines how much attention to pay to other words when forming a new representation of a word.
  - **Key Vector:** Is used to match with a query to compute attention.
  - **Value Vector:** Contains the information from the original word vector that we want to keep if the word is attended to.



SELF ATTENTION - KEY STEPS EXPLAINED

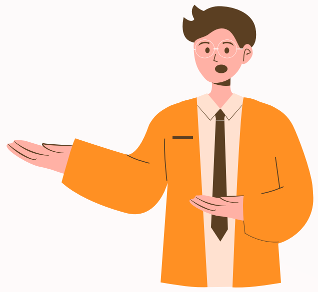
Scoring:

- **Dot Product:**
  - This is a mathematical operation that multiplies corresponding elements of two vectors and sums the results.
  - It’s a measure of similarity; the higher the dot product, the more similar the vectors.
- **Score**:
  - The result of the dot product between a query vector and a key vector.
  - It represents how much attention one word should pay to another.
  - A higher score means more attention.



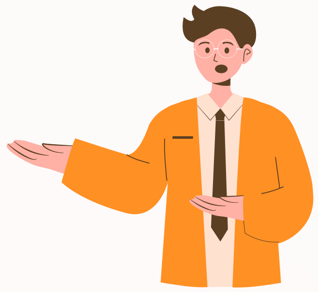
Scaling:

- **Dimension of the Key Vectors:**
  - This is the size of the key vectors (e.g., if the vector is [0.2, -0.4], the dimension is 2).
- **Square Root:**
  - The square root of the dimension is used to scale down the scores.
  - This prevents the softmax step from having extremely small gradients, which can slow down learning or lead to numerical instability.



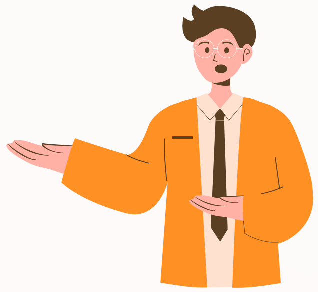
Softmax:

- **Softmax Function:**
  - A mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector.
- **Attention Weights:**
  - The output of the softmax function.
  - These weights sum up to 1 and indicate the importance of each word's value vector in the context of the other words.



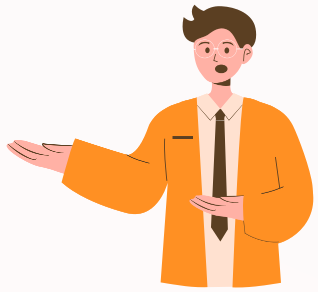
Weighted Sum:

- **Weighted Sum:**
  - This is the sum of the value vectors, each multiplied by its corresponding attention weight.
  - This step combines the information from all the value vectors into a single vector, taking into account the attention weights.



Output:

- **Output Vector:**
  - The final vector produced by the self-attention mechanism for each word.
  - It's a new representation of the word that incorporates information from other relevant words in the sentence.



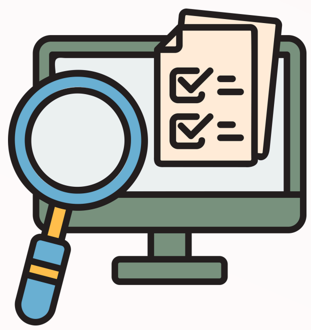
AN EXAMPLE

Let's consider a sentence with five words to explain the self-attention mechanism using two-dimensional vectors.

Our example sentence will be:
"Smart robots perform complex tasks."

Vector Representation: We start by representing each word with a two-dimensional vector. For simplicity, let's assign arbitrary vectors:

- "Smart" = [1, 0]
- "robots" = [0, 1]
- "perform" = [1, 1]
- "complex" = [2, 0]
- "tasks" = [0, 2]



AN EXAMPLE

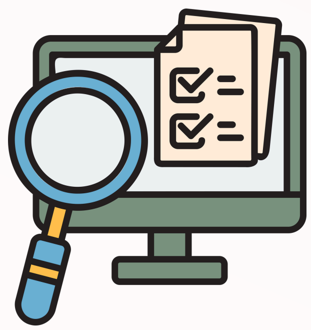
- **Query, Key, and Value Matrices:**
  - We have matrices Q, K, and V that transform the word vectors into query, key, and value vectors.
  - For this example, let's assume these matrices are the **identity matrix**, so the transformed vectors remain the same.

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

AN EXAMPLE

Dot Product Scores: We calculate the dot product of the query vector of "**robots**" with the key vectors of all words:

- Dot product of "robots" with "Smart"
  $$= [0, 1] \bullet [1, 0] = 0$$
- Dot product of "robots" with "robots"
  $$= [0, 1] \bullet [0, 1] = 1$$
- Dot product of "robots" with "perform"
  $$= [0, 1] \bullet [1, 1] = 1$$
- Dot product of "robots" with "complex"
  $$= [0, 1] \bullet [2, 0] = 0$$
- Dot product of "robots" with "tasks"
  $$= [0, 1] \bullet [0, 2] = 2$$



AN EXAMPLE

- **Scaling:** We scale the scores by dividing by the square root of the dimension of the key vectors, which is (\sqrt{2}) in this case:
    - Scaled score for “Smart” = 0 / (\sqrt{2}) = 0
    - Scaled score for “robots” = 1 / (\sqrt{2}) = 0.71
    - Scaled score for “perform” = 1 / (\sqrt{2}) = 0.71
    - Scaled score for “complex” = 0 / (\sqrt{2}) = 0
    - Scaled score for “tasks” = 2 / (\sqrt{2}) = 1.41

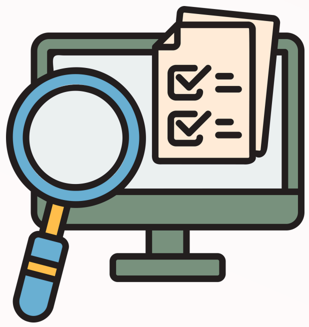
AN EXAMPLE

- **Softmax:**
  - We apply the softmax function to the scaled scores to get the attention weights:
  - Softmax(0, 0.71, 0.71, 0, 1.41) = [0, 0.19, 0.19, 0, 0.62] (approx)

*Softmax function takes a vector of numbers and squishes them into probabilities between 0 and 1, where all the probabilities add up to 1.
Think of it as turning raw scores into a confidence distribution for multiple classes.

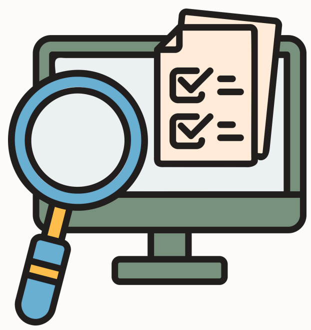
AN EXAMPLE

- **Weighted Sum:** Multiply the value vectors by the attention weights and sum them up:
  - Weighted sum = 0 * [1, 0] + 0.19 * [0, 1] + 0.19 * [1, 1] + 0 * [2, 0] + 0.62 * [0, 2] = [0.19, 1.43] (approx)



AN EXAMPLE

- **Output:** The output vector for the word "robots" after applying self-attention is approximately [0.19, 1.43].
- Similarly matrix can be calculated for rest of the words
    - "Smart": [1.28, 0.24]
    - "Perform": [0.87, 0.87]
    - "Complex": [1.64, 0.12]
    - "Tasks": [0.12, 1.64]



AN EXAMPLE

Here's the mathematical representation of the steps:

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

THE TRANSFORMER ARCHITECTURE

- Self-attention allows the model to weigh the importance of each word in the context of the entire sentence.
- When a word appears in different contexts or with different neighboring words, the attention mechanism can assign different weights to it, leading to different representations (embeddings) of the word in those contexts.
- The calculation in this document explained this in detail.
- With Post 14,15,16 and this covering some improtant aspects of transformer architecture, we are reaching close to understanding it end to end

Thank You

SPECIAL THANKS TO CHATGPT, OPEN AI, COPILOT, GEMINI FOR THE SUPPORT ON CONTENT

in DINESHLAL