Attention

- key mechanism for accessing relevant information in a context to make predictions

* How can attention impact language modeling?

Ex. Fixed length sequences of A and B

AAAAAAA
ABAAAA B
ABAABA B
AAAABA B

All A's : Last letter is A
Any B : Last letter is B

- Keys and Query

A A B A ___ (B)

Keys : embeddings of sequence
Query : what we want to find

Assume : $A = [1, 0]$, $B = [0, 1]$ } one-hot encoding embeddings e_i

- Step 1 : Compute score for each key given query

	[1 0]	[1 0]	[0 1]	[1 0]
	A	A	B	A
Dot product	0	0	1	0

score $s_i = K_i^T q$

set $q = [0\ 1]$ to find B's

- Step 2 : Softmax

A A B A

0 0 1 0 $\longrightarrow$ softmax : (assume $e = 3$) $[1/6 \quad 1/6 \quad 1/2 \quad 1/6]$ α_i

- Step 3 : Compute output as weighted sum of input

$$\text{result} = \sum_{i=1}^{4} \alpha_i e_i = \left(\tfrac{1}{6}[1\ 0] + \tfrac{1}{6}[1\ 0] + \tfrac{1}{2}[0\ 1] + \tfrac{1}{6}[1\ 0] \right)$$
$$= [1/2 \quad 1/2]$$

* Can make attention more peaked by amplifying the embeddings

$k_i = W_k \, e_i$ $\qquad W_k = \begin{matrix} 10 & 0 \\ 0 & 10 \end{matrix}$ $\qquad \begin{array}{cccc} A & A & B & A \\ [10\ 0] & [10\ 0] & [0\ 10] & [10\ 0] \\ 0 & 0 & 1 & 0 \end{array}$

- Original Dot product attention : $s_i = k_i^T q$
- Scaled Dot product attention : $s_i = k_i^T W q$
- Equivalent to having two weight matrices : $s_i = (W^k k_i)^T (W^q q)$

Self Attention

* Mechanism in transformer that processes entire sequence at once
* Every word in sentence is both a key and query simultaneously

Q : sequence length $\times$ d matrix (d = embedding dimensions = 2 for this example)
K : sequence length $\times$ d matrix

$W^Q = \begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix}$ (no matter the value, we're going to look for B's)

$W^K = \begin{matrix} 10 & 0 \\ 0 & 10 \end{matrix}$ "Booster" as before

$$E = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Q = EW^Q = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$$

$$K = EW^K = \begin{pmatrix} 10 & 0 \\ 10 & 0 \\ 0 & 10 \\ 10 & 0 \end{pmatrix}$$

<u>Scores</u>: $S = QK^T$ $\qquad S_{ij} = q_i \cdot k_j$

$$S = \begin{pmatrix} 0 & 0 & 10 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 10 & 0 \end{pmatrix}$$

* rows represent attention scores (first row: I care about word 3)

→ row-wise softmax turns it into a distribution per row (A)

<u>Output</u>: AE
(actually $A(EW^V)$)

- <u>Vasuani et al.</u>

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

$$Q = EW^Q, \quad K = EW^K, \quad V = EW^V$$

* Normalizing by $\sqrt{d_k}$ helps control scale of softmax, makes it less peaked

- <u>What does self attention produce?</u>
  - Square attention matrix (A) $\times$ input = same dimension as input
  - Computes a contextualized encoding for each word, preserving length of sequence

* <u>"The Illustrated Transformer"</u> by Alammar

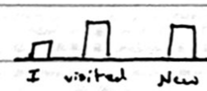
↳ Visualizes all of this

- **<u>Multihead Self Attention</u>**

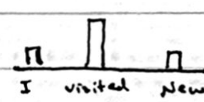
* Attention can theoretically learn to attend to multiple tokens, but in practice, softmax distributions become peaked:

I visited New _____

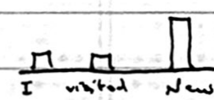
we want:



we get:



or



* <u>Solution</u>: Multiple heads to do independent copies of attention

↳ <u>Alammar for visualization</u>