Attention Mechanism: Mathematical Formulas

1

Compute Attention Scores

Similarity

$$e_{t,i} = \mathrm{score}(s_t, h_i) = s_t^T W_a h_i$$

Calculate similarity between decoder state s_t and each encoder hidden state h_i using learnable weight matrix W_a

2

Compute Attention Weights

Softmax

$$\alpha_{t,i} = \exp(e_{t,i}) / \Sigma_j \exp(e_{t,j})$$

Normalize scores using softmax to obtain attention weights that sum to 1. Higher scores → higher attention weights

3

Compute Context Vector

Weighted Sum

$$c_t = \Sigma_i \alpha_{t,i} \times h_i$$

Weighted sum of encoder hidden states using attention weights. More relevant states contribute more to the context

4

Generate Output

Prediction

$$\hat{y}_t = \mathrm{softmax}(W_o [s_t; c_t] + b_o)$$

Concatenate decoder state and context vector, then pass through output layer to predict next token

🔑 Key Components Summary

Input Dimensions

$s_t \in \mathbb{R}^d$

$h_i \in \mathbb{R}^d$

$W_a \in \mathbb{R}^{d \times d}$

Attention Properties

$\Sigma_i \ \alpha_{t,i} = 1$

$\alpha_{t,i} \in [0, 1]$

$0 \leq \alpha_{t,i} \leq 1$

Output

$c_t \in \mathbb{R}^d$

Dynamic context

Per time step

s_t: Decoder hidden state h_i: Encoder hidden state $e_{t,i}$: Attention score $\alpha_{t,i}$: Attention weight c_t: Context vector

W_a: Alignment weight matrix

🧮 Concrete Example with 3D Vectors

📝 **Setup:** d = 3 (vector dimension), using 2 encoder hidden states (h_1, h_2).

1 Given Input Data

Decoder hidden state (s_t):

$s_t = [1, 2, 1]^T$

Encoder hidden states:

$h_1 = [2, 0, 1]^T$

$h_2 = [1, 1, 2]^T$

Weight matrix (W_a) $\in \mathbb{R}^{3x3}$:

W_a = [1 0 0] [0 1 0] [0 0 1]

(Using Identity Matrix for simplicity)

2 Compute Attention Scores

📐 Formula: $e_{t,i} = s_t^T W_a h_i$

- **Score for h_1 :**

$e_{t,1} = s_t^T W_a h_1 = [1, 2, 1] \cdot [2, 0, 1]^T$

$= (1 \times 2) + (2 \times 0) + (1 \times 1)$

$= 2 + 0 + 1$

$= \mathbf{3}$

- **Score for h_2 :**

$e_{t,2} = s_t^T W_a h_2 = [1, 2, 1] \cdot [1, 1, 2]^T$

$= (1 \times 1) + (2 \times 1) + (1 \times 2)$

$= 1 + 2 + 2$

$= \mathbf{5}$

✓ Attention Scores:

$e_{t,1} = 3$, $e_{t,2} = 5$

3 Compute Attention Weights (Softmax)

📐 Formula: $\alpha_{t,i} = \exp(e_{t,i}) / \Sigma_j \exp(e_{t,j})$

- **Exponential calculation:**

$\exp(e_{t,1}) = \exp(3) \approx 20.09$

$\exp(e_{t,2}) = \exp(5) \approx 148.41$

- **Sum:**

$\Sigma \exp(e_{t,j}) = 20.09 + 148.41 = 168.50$

- **Attention Weights:**

$\alpha_{t,1} = 20.09 / 168.50 \approx$ **0.119**

$\alpha_{t,2} = 148.41 / 168.50 \approx$ **0.881**

- **Verification:**

$\alpha_{t,1} + \alpha_{t,2} = 0.119 + 0.881 = 1.000$ ✓

✓ Attention Weights:

$\alpha_{t,1} \approx 0.119$ (11.9%), $\alpha_{t,2} \approx 0.881$ (88.1%)

💡 **Interpretation:** h_2 receives about 88% attention because it's more similar to s_t!

4 Compute Context Vector

📐 Formula: $c_t = \Sigma_i \alpha_{t,i} \times h_i$

- **Weighted sum calculation:**

$c_t = \alpha_{t,1} \times h_1 + \alpha_{t,2} \times h_2$

- **First term:**

$\alpha_{t,1} \times h_1 = 0.119 \times [2, 0, 1]^T$

$= [0.238, 0.000, 0.119]^T$

- **Second term:**

$\alpha_{t,2} \times h_2 = 0.881 \times [1, 1, 2]^T$

$= [0.881, 0.881, 1.762]^T$

- **Final Context Vector:**

$c_t = [0.238, 0.000, 0.119]^T + [0.881, 0.881, 1.762]^T$

$= [1.119, 0.881, 1.881]^T$

🎯 Final Context Vector:

$c_t = [1.119, 0.881, 1.881]^T$

✨ **Result Interpretation:** The context vector is formed closer to h_2 (88.1% weight). This means the current decoder state is more relevant to h_2.

🎓 Key Takeaways

1. Attention Score measures similarity via dot product between two vectors
2. Softmax normalizes scores into probabilities between 0 and 1
3. Context Vector is a weighted average, focusing on important information
4. High similarity → high attention weight → greater influence