ML – Attention mechanism

Last Updated : 02 May, 2025

Attention mechanism helps models to focus on the most important parts of input data like humans prioritize certain information in a complex environment. It helps in improving models ability to perform tasks like language translation, image recognition and speech processing. In this article, we will see attention mechanism in detail, how it works and its application.

Understanding Attention Mechanism

Attention mechanism is a type of neural network that helps a model focus on specific parts of the input data, it is done by assigning weights to different elements in input which helps the model to decide which parts of information are most importart. This makes the model better at understanding complex relationships and dependencies in data. It helps the model to manage long-term dependencies and improves its ability to focus on important features.

For example, a recurrent neural network (RNN) picks different parts of image to explore over time. This method works better than the traditional convolutional neural networks (CNNs) for classification tasks and can be used in applications like robotics where it helps robots to decide how to act based on feedback from previous actions. This makes the system more flexible and able to learn and adapt over time.

How Attention Mechanism Works?

In a neural network model it works by:

1. Input Encoding: Input data is transformed into a format that the model can process and creating representations of the data.

2. Query Generation: A query vector is generated based on the current state or context of the model. This query tells the model what it is looking for in the input data.

3. Key-Value Pair Creation: Input is splitted into key-value pairs:

- **Keys** represents the important information required to measure the relevant data
- **Values** hold actual data.

4. Similarity Computation: Model calculates similarity between the query vector and each key. This helps find how relevant each part of the input is. Various methods can be used to calculate this similarity such as dot products or cosine similarity.

$$Score(s,i) = \begin{cases} h_s^{(1)} \cdot y_i & \text{Dot Product} \\ (h_s^{(2)})^T W y_i & \text{General} \\ v^T \tanh\left(W \begin{bmatrix} h_s \\ y_i \end{bmatrix}\right) & \text{concat} \end{cases}$$

where

- hs: Encoder source hidden state at position s
- yi: Encoder Target hidden state at the position i
- W: Weight Matrix
- v: Weight vector

5. Attention Weights Calculation: Similarity scores are passed through a softmax function to find attention weights. These weights indicate the importance of each key-value pair.

$$\text{Attention Weight } (\alpha(s,i)) = \text{softmax}(\text{Similarity Scores}(s,i))$$

6. Weighted Sum: Attention weights are applied to the corresponding values which helps in generating a weighted sum. This step adds the relevant information from the input based on their importance calculated by the attention mechanism.

$$c_t = \sum_{i=1}^{T_s} \alpha(s,i) h_j^{(1)}$$

Here

- T_s: Total number of key-value pairs (source hidden states) in the encoder.

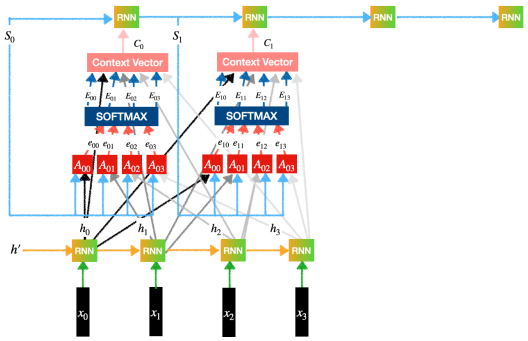
7. Context Vector: Weighted sum act as a context vector which represents attended information from the input. It captures the relevant context for the current task.

8. Integration with the Model: Context vector is combined with the model's current state or hidden representation which provides additional information for other steps of the model.

9. Repeat: Steps 2 to 8 are repeated for each step of the model which allows the attention mechanism to focus on different parts of the input sequence or data.

Attention Mechanism Architecture for Machine Translation

Attention mechanism in machine translation has three main components: **Encoder**, **Attention** and **Decoder**. Here's how each of these components works:

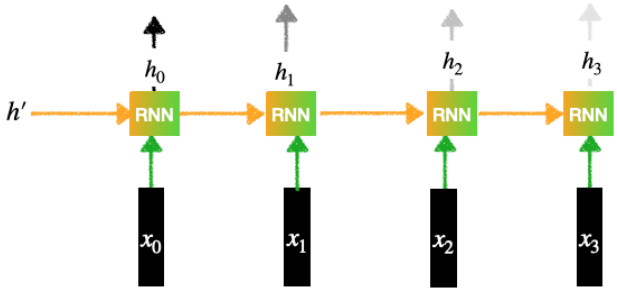


Encoder-Decoder with Attention

1. Encoder:

Encoder processes the input sequence which is usually a sentence and generates hidden states. It uses Recurrent Neural Networks (RNNs), LSTMs, GRUs or transformer-based models to process the input step by step. At each step encoder produces a hidden state that contains both the data from the previous hidden state and the current input token. This allows the encoder to capture the relationships between the tokens in the input sequence. As the input sequence is processed encoder generates a series of hidden states which together represent the entire input sequence. These hidden states are passed on to the attention mechanism for further processing.

It is important because it creates a representation of the input sequence that the attention mechanism and decoder will later use to generate an accurate output sequence.



Encoder

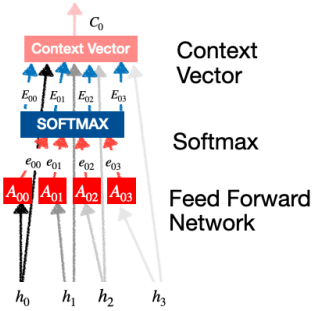
Contains an RNN layer (Can be LSTMs or GRU):

- Let's say, there are 4 words sentence then inputs will be: x_0, x_1, x_2, x_3
- Each input goes through an Embedding Layer, It can be RNN, LSTM, GRU or transformers
- Each of the inputs generates a hidden representation.
- This generates the outputs for the Encoder: h_0, h_1, h_2, h_3

2. Attention:

Attention component find importance of each encoder's hidden state with respect to the current target hidden state. It generates a context vector that captures the relevant information from the encoder's hidden states. Its mechanism can be represented mathematically as follows:

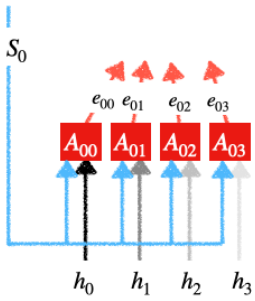
- Our goal is to generate the context vectors.
- For example, context vector C_1 tells us how much importance, attention should be given the inputs: x_0, x_1, x_2, x_3 .
- This layer in turn contains 3 subparts:
  1. Feed Forward Network
  2. Softmax Calculation
  3. Context vector generation



Attention

Feed Forward Network:

It transform target hidden state into a form that is compatible with the attention mechanism. This transformation is done using a linear transformation followed by a non-linear activation function like ReLU or sigmoid. This step helps in finding alignment between target and the encoder's hidden states.



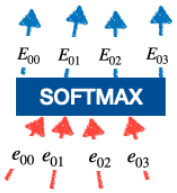
Feed-Forward-Network

- Previous Decoder state
- The output of Encoder states.

Each unit generates outputs: $e_{00}, e_{01}, e_{02}, e_{03}$ i.e $e_{0i} = g(S_0, h_i)$. Here **g** can be any activation function such as sigmoid, tanh or ReLu.

Softmax Calculation:

Once the feed-forward network generates the relevant scores these scores are passed through a softmax function. It converts scores into probability-like attention weights. These weights indicate the relative importance of each encoder's hidden state in generating current target token. For example higher attention weights shows that a particular encoder's hidden state is more important for the current step in the decoding process.



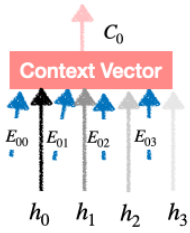
softmax calculation

$$E_{0i} = \frac{exp(e_{0i})}{\sum_{i=0}^{3} exp(e_{0i})}$$

These $E_{00}, E_{01}, E_{02}, E_{03}$ are called the attention weights. It decides how much importance should be given to the inputs x_0, x_1, x_2, x_3 .

Context Vector Generation:

When the attention weights are calculated they are applied to encoder's hidden states which helps in generating a weighted sum. This weighted sum forms the **context vector** which finds most relevant information from the encoder's hidden states based on their attention weights.



context vector generation

$$C_0 = E_{00} * h_0 + E_{01} * h_1 + E_{02} * h_2 + E_{03} * h_3$$

We find C_1, C_2, C_3 in the same way and feed it to different RNN units of the Decoder layer. So this is the final vector which is the product of Probability Distribution and Encoder's output which is nothing but the attention paid to the input words.

3. Decoder:

It receives the context vector from the attention layer which contains relevant information from the encoder's hidden states along with its own current hidden state. Using this combined information decoder predicts next token in the output sequence. Decoder's hidden state is then updated based on the predicted token and this process repeats again and again for each token in the output sequence. This helps model to generate entire output sequence like translation word by word with each step focusing on the most important parts of the input.

Applications of attention mechanisms

1. **Machine Translation**: It allows model to focus on different parts of the source sentence when generating each word in the target sentence which improves translation accuracy.
2. **Sentiment Analysis & Named Entity Recognition**: In tasks like sentiment analysis, question answering and named entity recognition it helps models to focus on key words that helps in entity identification.
3. **Text Summarization**: It helps in selecting important information by helping model to create short and accurate summaries of large text.
4. **Image Captioning**: It help image captioning models to focus on specific image part while generating captions which improves description's detail.

Attention mechanism will keep getting better and making models more efficient and useful. This will help them perform well in more tasks and handle even more complex problems.

Next Article

Types of Attention Mechanism

Similar Reads

Multi-Head Attention Mechanism

The multi-head attention mechanism is a key component of the Transformer architecture, introduced in the seminal paper "Attention Is All You Need" by Vaswani et al. in 2017. It plays a crucial role in enhancing the ability of models to focus on different parts of an input sequence simultaneously, ma

3 min read

Types of Attention Mechanism

Attention mechanisms are crucial in deep learning, helping models perform better in tasks like NLP and computer vision. They enable models to focus on important parts of the input data, much like how humans concentrate on key details while ignoring irrelevant information helping in better understand

7 min read

Transformer Attention Mechanism in NLP

Transformer model is a type of neural network architecture designed to handle sequential data primarily for tasks such as language translation, text generation and many more. Unlike traditional recurrent neural networks (RNNs) or convolutional neural networks (CNNs), Transformers uses attention mech

6 min read

Cross-Attention Mechanism in Transformers

Cross-attention mechanism is a key part of the Transformer model. It allows the decoder to access and use relevant information from the encoder. This helps the model focus on important details, ensuring tasks like translation are accurate. Imagine generating captions for images (decoder) from a deta

5 min read

Attention Mechanisms for Computer Vision

Attention mechanisms have revolutionized the field of computer vision, enhancing the capability of neural networks to focus on the most relevant parts of an image. By dynamically adjusting the focus, these mechanisms mimic human visual attention, enabling more precise and efficient processing of vis