Understanding the Attention Block!

Aniket Phutane

January 12, 2022
RWTH Aachen University

1 Introduction

The attention mechanism was first introduced in the context of Computer Vision. As humans, when we are exposed to an image, we have the tendency to roughly scan the image. Eventually, we focus on some regions of interest and then identify the object. On identification, we focus on the location of the object.

In Computer Vision, attention is used to highlight important parts of an image that contributes to the desired output. For example, if we want to identify the position of a building in an image, we use attention to essentially see which pixels are aligned with the concept of a building and then the heat-map corresponds to the weights of the attention mechanism. In a general case, we have weights over all the pixels in an image. Then, we try to see which pixels would have some semantic meaning or embedding which aligns with the notion of the object we're trying to recognize.

It was first introduced in the Natural Language Processing space in 2015 for solving the Machine Translation problem. Machine translation is the process of using artificial intelligence (AI) to automatically translate content from one language (the source) to another (the target) without any human input. With the help of this mechanism, we could get our model to essentially peak/look back at the input sentence without losing track of what it is translating. Also, it need not remember the complete input sentences. This was not possible with the previous machine translation problem-solving variants. It was considered a very important breakthrough as it allowed us to deal with sentences of arbitrary lengths.

In 2017, this idea was expanded for the Language Modelling problem. For this problem, we develop a model to predict/recover the next word in a sequence that could generate words. Most of the problems in NLP lie under the umbrella of Language Modelling Problems.

2 Attention better than RNN's

2.1 Large number of training steps

Recurrent neural networks are a type of neural network where the outputs from previous time steps are fed as input to the current time step. This creates a network graph or circuit diagram with cycles, which can make it difficult to understand how information moves through the network. The number of steps required for training recurrent neural networks is quite large. The optimization tends to be quite difficult as compared to the attention mechanism.

2.2 Parallel Computation

In today's world, GPUs are extensively used for working with neural networks. In Recurrent Neural Networks, computation has to be done sequentially. We can't process all the steps in

parallel because of the recurrence. As this is an inherent problem, we can't leverage GPUs in the context of RNN's. The attention mechanism allows us to do the computations in parallel with the help of GPUs.

2.3 No Gradient Vanishing and Explosion

Instead of having computations that goes linearly with the length of the sequence into a deep network for RNN, we do the computations for the entire sequence simultaneously in the attention mechanism.

3 Attention Mechanism - Intuitive Idea

3.1 Data Retrieval

In a database, if we want to retrieve a value based on a query, we identify the key that aligns well with the query and then simply output the corresponding value. From that standpoint, this mechanism can be considered as some form of an approximation of the 'SELECT' query we use for data retrieval.

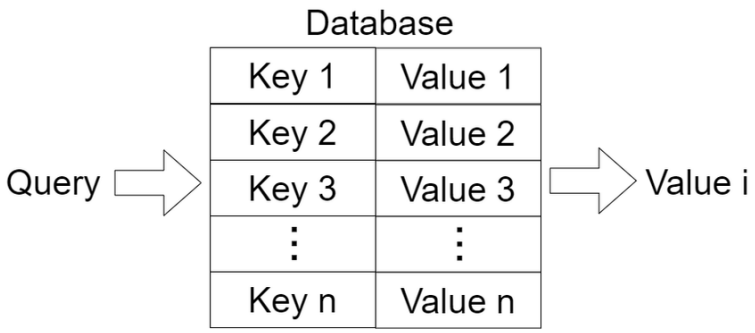


Figure 1: Data Retrieval In a Database.

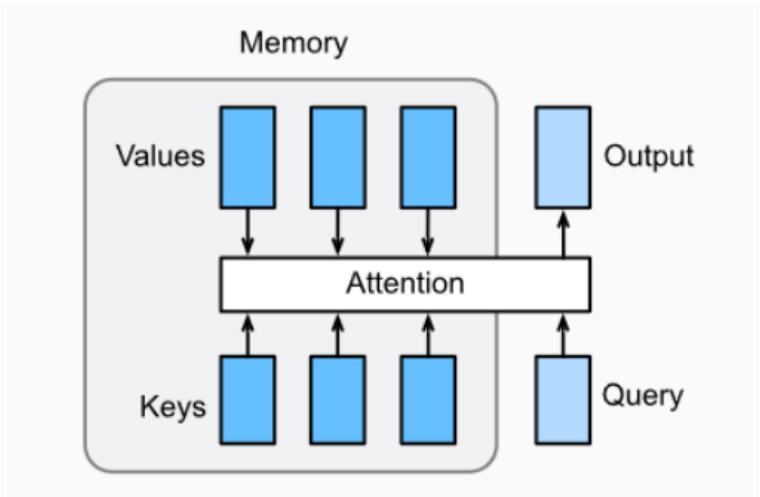


Figure 2: The Attention layer returns an output based on its input query and its memory

3.2 Similarity Measure

Attention mimics the retrieval of a **value** v for a **query** q based on a **key** k in a database.

$$attention(q, k, v) = \sum_i similarity(q, k_i) \times v_i$$

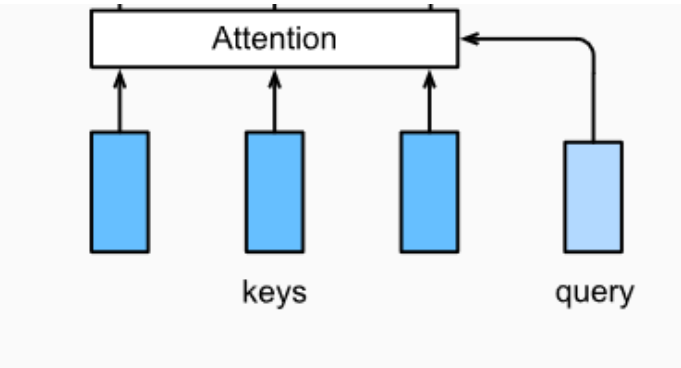
The similarity function measures the similarity between query q and each key-value ki. This returns a weight for every query-key pair and then we produce an output that is a weighted combination of all the values in our database.

In a database when we do a retrieval, we simply return one value that corresponds to the respective key. Here, this corresponds to finding a similarity between the query and some key where the similarity measure would have value '1' and for all the other keys, the similarity measure would return the value '0'. The similarity function essentially produces a one-hot encoding setup and we effectively return one value.

In practice, we want to embed this encoding as a part of a neural network and be able to do back-propagation for retraining the weights. The similarity measure can be considered as computing a distribution with weights between 0 and 1. If we have multiple keys that have some similarity, the idea is to produce a value that is a weighted combination based on the weights.

In summary, the retrieval process is a weighted combination of the values for which the keys have high similarity in a database.

4 Neural Architecture



In the first layer, we compute the similarity measure between the query and each key.
$s_i = f(q, k_i)$
The function f can be a

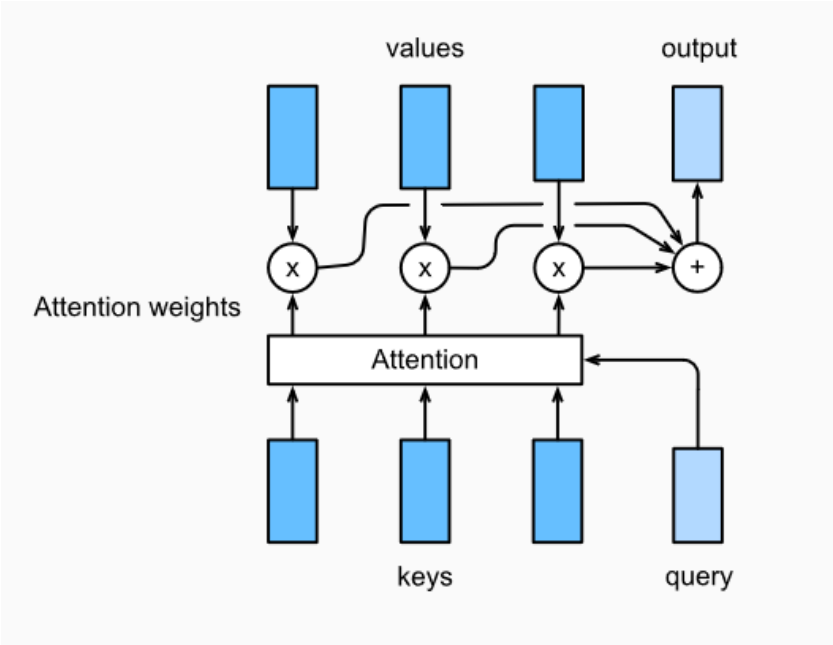
1. Dot Product between the query-key pair : $q^t \cdot k_i$.
2. Scaled dot product : $q^t \cdot k_i/\sqrt{d}$ where d is the dimensionality of each key.
3. General dot product : $q^t \cdot W \cdot k_i$ where W can be used as a weight matrix for transforming the queries to be in the same space as the keys.
4. Additive similarity : $w_q^T \cdot q + w_k^T \cdot k$

We compute the attention weights using the softmax function. The softmax function takes as input a vector z of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers.

This is given by

$$a_j = exp(s_i)/\sum_i exp(s_i)$$

$s_i \times a_j$ is a fully connected network.



Finally, we compute the attention value by performing an element-wise multiplication and addition between the obtained attention weights and values.

$$\sum_i a_i \cdot v_i$$

The output is a linear combination of values and the attention weights which are obtained from the notion of similarity between the query and the keys.

References

[1] N. P. J. U. L. J. A. N. G. K. Ashish Vaswani, Noam Shazeer, "Attention is all you need," 2017.

[2] C. Bishop, in *Pattern Recognition and Machine Learning*, 2006.