Keys, Values, Queries

‘Bank of the River’ - each word is tokenized into a token t

Each token t has a corresponding WE, V.

They are pushed into the self-attention block and we expect the output, y, to be context aware.

To get context of the WE V3, we take its dot product with all the other vectors.

V3V1, V2V3, V3V3, V4V3…..

Let’s call the output of these dot product operations as Scores.

Normalize the scores so that things don’t go out of hand – Out comes the Weights Wij

All the weights sum to 1, since they have been normalized.

We started with vectors Vx and now we have weights or numbers, Wij.

We take the normalized weights and multiply with the WE Vx again to get the more contextualized out Yn.

Yn is again a vector, but a contextualized ‘weighted’ representation of the input text.

Note that this system right now has no weights, so it is not really learning anything except from whatever help it can get from the WE.

We can add weights to make sure that the system learns even more latent patterns.

The weights can be added where we are having operations with the WE.

The Database Analogy:

It can be said that the WE for which we are trying to find the context, is the *QUERY.*

The WE of all the other words are *KEYS,* and the final multiplication of the normalized weights with the WE is *VALUES.*

Adding Trainable Weights:

We need to add weights that can be trained, for the whole system here.

A smart way to do that is to multiply Vi with a matrix M.

Vi dimensions are *1 x k* and M’s dimensions would be *k x k.*

Dimensions of the resultant vector will be *1 x k*.

Now we can multiply all the vectors with the keys matrix MK, and query vector with the matrix MQ and the value vectors with the value matrix MV.

With these matrices acting as trainable weights/parameters, we can now ‘train’ our *Attention Block.*

*Interesting point to note at this stage is the fact that all these operations are kind of independent of each other.*

Now we can push each Word Embedding to a Linear Layer, one for each Keys Queries and Values, and then combine the output. All of this happens in parallel.

The output is a collection of *Contextualized Word Embeddings*.

We can use this block over and over again in a neural architecture.

Note that the values of the Word Embeddings are not being modified by the Neural Architecture, it is only modifying the matrices MK MQ and MV.