Attention Mechanism:

Time Series Example:

TS is noisy -> So we can re-weigh the points to reduce the noise

We are re-weighing a particular point in the TS in order to amplify it, and suppress the rest of the points.

We can do that by multiplying the TS by a re-weighing factor, Wi.

Wi is *Normally Distributed*.

We simply take the dot product of the TS with Wi to reweigh.

We are essentially sliding Wi across the TS. Wi is acting as a filter on TS.

New TS can be postulated to be more useful since the data points gave CONTEXT to each other.

How we do this for text data?

For Text Data:

It can be said that the words closest to a particular word carry the highest importance.

But this is not a great strategy for ‘human language’.

Human Languages are COMPLEX.

In comes Word Embeddings!

Let’s say each token has an associated vector or word embedding vector with it, Vi.

We have a re-weighing method/scheme with W1, W2, W3…….Wn

We multiply Vi with Wi to get a more context aware version Yi.

Word Embeddings carry the latent meaning of a particular word.

They encode the essence of every word.

Each number in a word embedding carries some information about how that particular word is related with other words.

Essentially, Word Embedding vectors contain how ‘similar’ a word is with some other word, irrespective of the distance between them in a sentence.

Similar words will have similar word embeddings.

Hence, we can use Word Embeddings to ‘re-weigh’ our text.

Now we can get better context using word embeddings in our re-weighing scheme.

We multiply the word embedding vector with the WE vector of every other word and obtain Wxy.

Wxy is ‘normalized’ = They all sum to 1

Important points to note here:

1. No training of weights
2. Order has no influence
3. Proximity has no influence
4. Shape independent – we can have as much text as we want