**Seq2Seq Model Building Theory and Research**

**Training Our Base Model : -**

* We have questions and answers in the dataset.
* We required some kind of neural network to predict the next word.
* To build a neural network we required 1’s and 0’s.
* We can use “**One Hot Encoding**”.
  + Very long string
  + Somewhere in the middle there is 1 present
  + Dataset requires 7910 neurons
  + very inefficient
* Efficient way “**Embedding Matrix**”.
  + set of word-vectors
  + vectors close together in N-dimensional space have similar meanings based on context, and word-vectors distant to each other have differing meanings.
  + Example :- **strong** and **powerful** close together where, **Strong** and **London** relatively far together.
  + We required 200 neurons to implement this kind of matrix.
* **First layer** of the **neural network** would be the “**Embedding Layer**”.
* We required our model to understand the relation between text and hence for it **LSTM** and **GRU** layers came into picture.
* We have to capture contexts of our questions and answers independentlyand then somehow **tie these contexts together**.
* We can use **Two LSTM** layers simultaneously :-
  + Train the **First LSTM layer** one on **questions**.
  + Use its **initial weights** to train the **second LSTM** layer on **answers**.
  + **First layer will act as an encoder** — it will process the input sequence and return its own internal state.
  + This state will serve as the **context** of the **second LSTM layer — decoder**.
  + Decoder will be trained to predict the next words of the target sequence, given the previous words.
  + Effectively, the decoder learns to generate **targets[t+1…]** given **targets[…t]**, conditioned on the input sequence.
* The **dense** layer will predict the confidence score for each word and this output will be used to evaluate the quality of the training process.
* We can apply **softmax** to obtain the probabilities and then use **categorical crossentropy** loss function to calculate the loss.

**Summary of Base Model : -**

1. Input layer One - For Questions. (**E\_Input\_1**)
2. Input layer Two - For Answers. (**D\_Input\_2**)
3. Embedding layer One - For Questions. (**E\_Embedding\_1**)
4. Embedding layer Two - For Answers. (**E\_Embedding\_2**)
5. LSTM - For Questions. (**E\_LSTM\_1**)
6. LSTM - For Answers. (**D\_LSTM\_2**)
7. Dense layer (**Dense**)

**Built Model To Decode the User Input Sequence : -**

* This model is used to **predict the user query answer** : -
  + Encode the input sequence into state vectors.
  + Start with a target sequence of size 1 (just the word start in our case)
  + Feed the state vectors and 1-word target sequence to the decoder to produce predictions for the next word.
  + Sample the next word using these predictions.
  + Append the sampled word to the target sequence.
  + Repeat until we generate the end-of-sequence word end or we hit the length of the answer limit.

1. **Inference Encoder : -**
   * For the **inference encoder**, the **input** is a **single question** represented as a **sequence** of **integers** padded with zeros and the **output** is just an **internal state vector** of the **LSTM layer.**

* **Summary of Inference Encoder : -**

1. Input layer One - For input Question. (**E\_Input\_1**)
2. Embedding layer One - For input Questions. (**E\_Embedding\_1**)
3. LSTM - For input Questions. (**E\_LSTM\_1**)
4. **Inference Decoder : -**
   * We have **3 input layers**.
     + 1-word target sequence
     + Next two for the **state vectors**, returned by the **Inference Encoder.**
   * Only the **first input** is connected to the **Embedding layer,** because we need to **encode** the **target sequence** as an **embedding vector.**
   * **Output** of the **decoder** is the **next predicted** word in **one-hot encoding.**
   * **Summary of Inference Decoder : -**
5. Input layer One - For 1 word generation (**D\_Input\_2**)
6. Embedding layer One - For input answers. (**D\_Embedding\_2**)
7. Two state vector layers return by Inference encoder. (**D\_Input\_3**,**D\_** **Input\_4**)
8. LSTM - For input Questions (**D\_LSTM\_2**).
9. Dense layer (**Dense**)