**Q.** ***The aim of this challenge is to create a machine learning model that can translate from English to German. Can you convince C-suite of an international company that your custom solution trained on company specific data is better than of-the-shelf solutions from big internet companies? Can you think of possible reasons why a company would hesitate to use a solution from big internet company and prefer an in-house solution? Use them together with results from your model to convince them to use your solution***

**The objective is to convert an English sentence to its German counterpart using a Neural Machine Translation (NMT) system. I divided my assignment into different notebooks. If all work is done in one file, then it may get very memory heavy and it is not easy to make changes. I will explain content and role of each notebook briefly.**

# **1-Work 0:**

# **2-Combine:**

# **3-Preprocessing:**

The data we work with is more often than not unstructured so there are certain things we need to take care of before jumping to the model building part.

***3.1)-Text Cleaning***

* get rid of the punctuation marks and then convert all the text to lower case.

***3.2)-Text to Sequence Conversion***

* A Seq2Seq model requires that we convert both the input and the output sentences into integer sequences of fixed length.
* vectorize our text data by using Keras’s Tokenizer () class. It will turn our sentences into sequences of integers. We can then pad those sequences with zeros to make all the sequences of the same length.

# **4-Applied Models**

RNN:

Recurrent Neural Networks (or more precisely LSTM/GRU) have been found to be very effective in solving complex sequence related problems given a large amount of data. They have real time applications in speech recognition, Natural Language Processing (NLP) problems, time series forecasting, etc

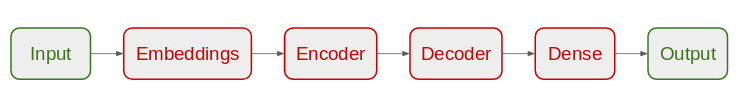
Sequence-to-Sequence:

Sequence to Sequence (often abbreviated to seq2seq) models are a special class of Recurrent Neural Network architectures typically used (but not restricted) to solve complex Language related problems, such as natural language translates, text summarization, speech recognition, chatbot, among others. Our aim is to translate given sentences from one language to another. Therefore, we used seq2seq model for our problem.

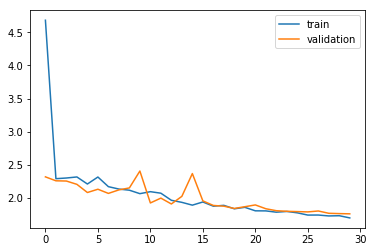
## ***4a)-Simple Model:***

**This model is like basic language model using a typical seq2seq model. There are two major components i.e encode & decoder.** Both these parts are essentially two different recurrent neural network (RNN) models combined into one giant network. Our simple model architecture consists of following;

* For the encoder, we will use an embedding layer and an LSTM layer
* For the decoder, we will use another LSTM layer followed by a dense layer



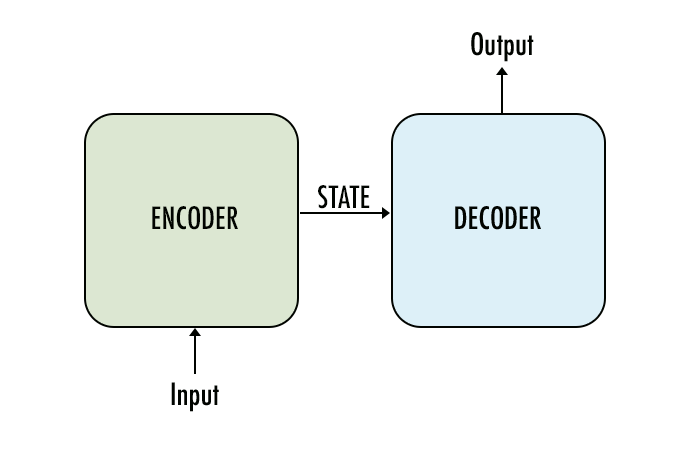
**This model gave not very impressive results implying that we may need to develop a better model.**

****

we have used ‘sparse\_categorical\_crossentropy‘as the loss function. This is because the function allows us to use the target sequence as is, instead of the one-hot encoded format**. One-hot encoding the target sequences using such a huge vocabulary might consume our system’s entire memory.**

## ***4b)-Encoder-decoder model***

This model is classical encoder-decoder model as per literature. In this model, we will break the sentence by words as this scheme is more common in real world applications using greedy approach. We will select the next word using the highest probability in the softmax layer.



* In our model, both encoder and the decoder are LSTM models.
* Encoder reads the input sequence and summarizes the information in something called as the internal state vectors (in case of LSTM these are called as the hidden state and cell state vectors). We discard the outputs of the encoder and only preserve the internal states.
* Decoder is an LSTM whose initial states are initialized to the final states of the Encoder LSTM. Using these initial states, decoder starts generating the output sequence.
* The decoder behaves a bit differently during the training and inference procedure. During the training, we use a technique call teacher forcing which helps to train the decoder faster. During inference, the input to the decoder at each time step is the output from the previous time step.
* Intuitively, the encoder summarizes the input sequence into state vectors (sometimes also called as Thought vectors), which are then fed to the decoder which starts generating the output sequence given the Thought vectors. The decoder is just a language model conditioned on the initial states.

## ***4c)- Applying out of vocabulary concept***

The difficulty of the translation task is proportional to the size of the vocabularies, which in turn impacts model training time and the size of a dataset required to make the model viable. We shall reduce the vocabulary of both the English and German text and mark all out of vocabulary (OOV) words with a special token(unk).

we can count the occurrence of each word in the dataset. For this we can use a *Counter* object, and updates a count each time a new occurrence of each word is added.

We can then process the created vocabulary and remove all words from the Counter that have an occurrence below a specific threshold. Some use 90 percentile. I have used threshold of 5.

***4d)-Applying sparse categorical accuracy as evaluation***

***5d)- Try attention model***

**5)-Further improvements**

* Get much more data. Top quality translators are trained on millions of sentence pairs.
* Build more complex models like Attention.
* Use dropout and other forms of regularization techniques to mitigate over-fitting.
* Perform Hyper-parameter tuning. Play with learning rate, batch size, dropout rate, etc. Try using bidirectional Encoder LSTM. Try using multi-layered LSTMs.
* Try using \*beam search instead of a greedy approach.
* Try BLEU score to evaluate your model.

The list is never ending and goes on.

\*Beam search= an approach in which we consider multiple words for a single input word and creates beam and thus creates multiple sentences while finally choosing the sentence which has the highest overall probability.