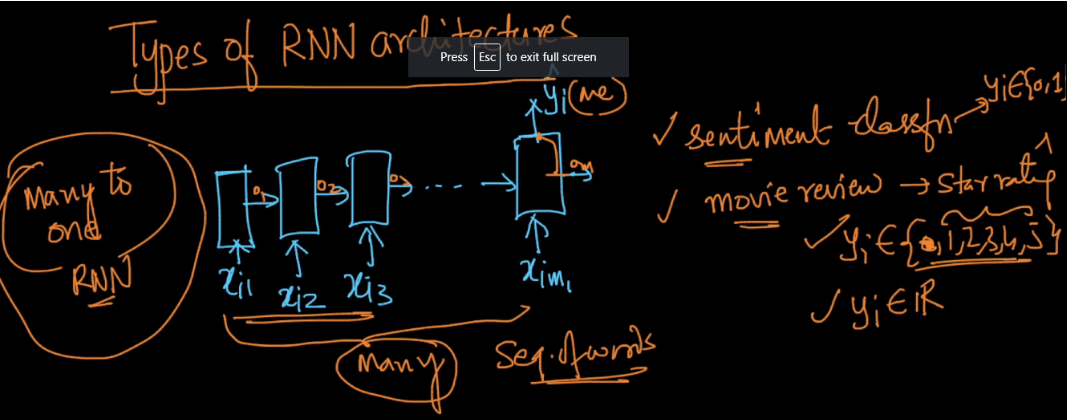
**Types of RNN architectures:**

1. Many to one
2. One to Many
3. Many to Many

**Many to One:**

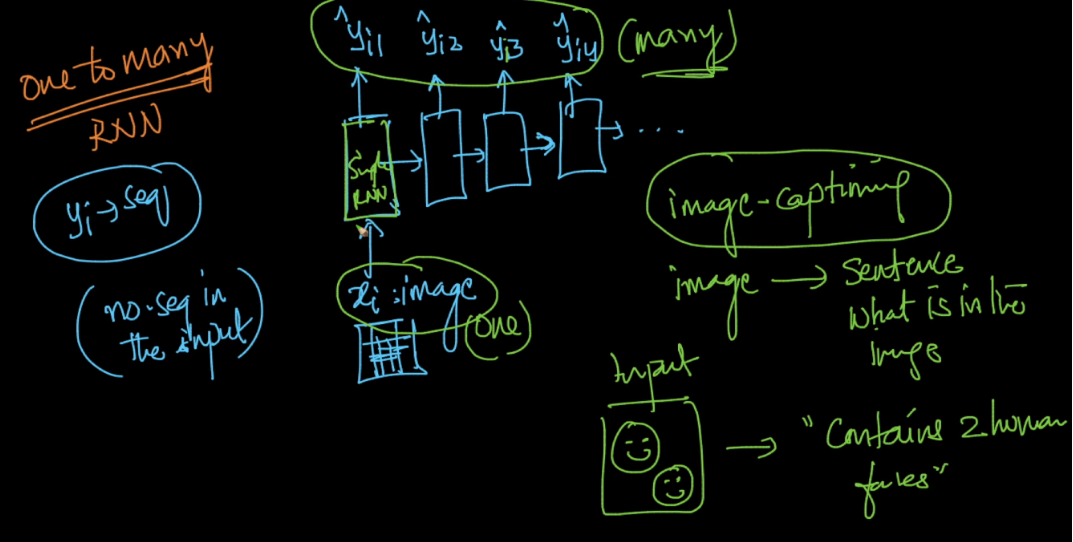
All RNN we seen till now are Many to one where there are multiple words in a sentence hence many input and only one output and hence one ouput.

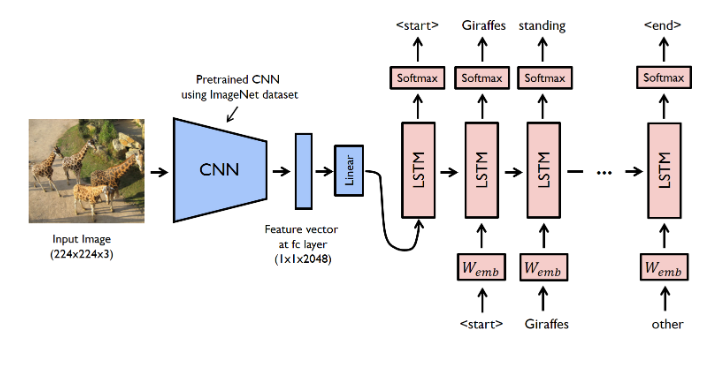
Example: Sentiment classification(0, 1), Movie Review(o/p b/w 1 to 5).



**One to Many:**

In such RNN we have only one input and many output, one standard example for this architecture is image-captioning where there is only one image as input and we need to generate a sentence(contains many words), each word generated one by one in different time step. Let’s a given image has 2 faces, so RNN will generate output as “contains 2 human faces”





**Many to Many RNN:**

There are 2 variants of many to many RNN.

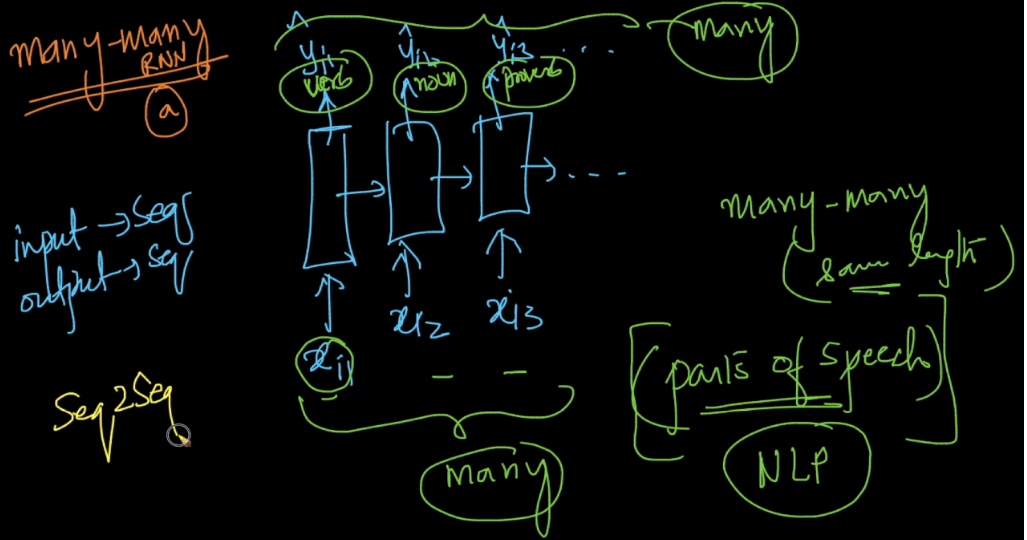
1. Input and output are of same length
2. Input and output are of different length

**Input and o/p are of same length:**

A example for such architecture is finding Parts of Speech of words in a sentence, so here for each word we need to find it’s POS hence many to many RNN.

Such architecture are also called **Sequence to Sequence RNN** because i/p and o/p both are sequence.

<https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html>



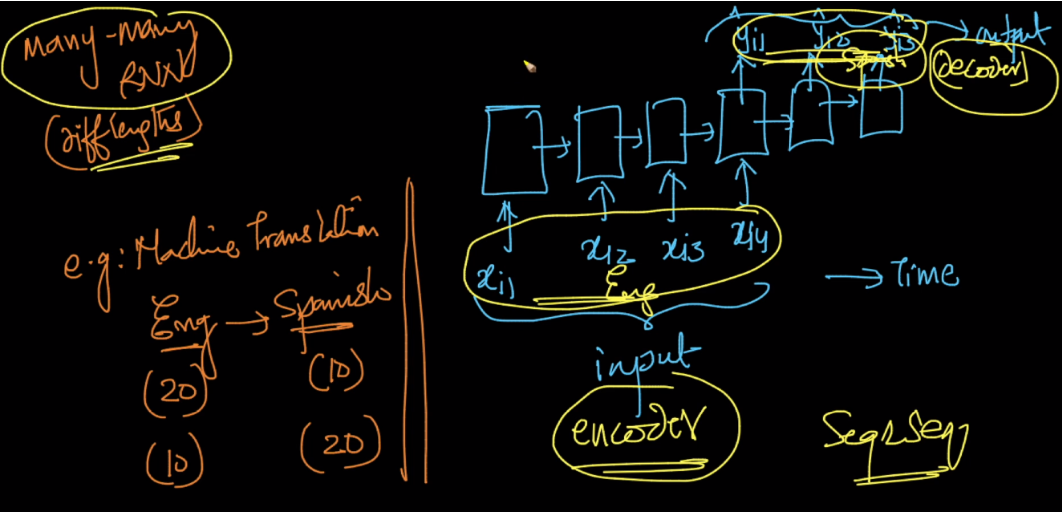
**I/p and o/p are of different length:**

One such ex for this architecture is Machine translation, let’s say a model need to convert English to Spanish, so there can situations where a sentence in English have 20 words and it’s Spanish equivalent have only 10 words in Spanish or vice versa.

So for such architecture there will be input provided first and once all words are provided as input then output units start one by one.

The units which consuming input are called encoder and units which giving outputs are called decoder.

Note: For a seq2seq Translation RNN model, we typically do not overlap the encoder and decoder units. Typically, y\_i1 will start as the output of a cell/unit after x\_i4 in this diagram and NOT as the output of the cell which inputs x\_i4. This would clearly separate encoder and decoder units



**We can say that there is one more type of RNN: One to One**

But this architecture is a simple MLP.

