**NLP- ASSIGNMENT-4**

1. **Can you think of a few applications for a sequence-to-sequence RNN? What about a sequence-to-vector RNN? And a vector-to-sequence RNN?**

Applications are speech recognition, machine translation, image captioning and question answering. A variable-length context vector can be used instead of a ﬁxed-size vector.

An Attention mechanism can be used to produces a sequence of vectors from the encoder RNN from each time step of the input sequence. The Decoder learns to pay selective attention to the vectors to produce the output at each time step.

A variable-length context vector can be used instead of a ﬁxed-size vector. An Attention mechanism can be used to produces a sequence of vectors from the encoder RNN from each time step of the input sequence. The Decoder learns to pay selective attention to the vectors to produce the output at each time step.

1. **Why do people use encoder–decoder RNNs rather than plain sequence-to-sequence RNNs for automatic translation?**

The encoder-decoder architecture for recurrent neural networks is the standard neural machine translation method that rivals and in some cases outperforms classical statistical machine translation methods.

1. **How could you combine a convolutional neural network with an RNN to classify videos?**

Each video is converted into sequential images and passed onto the CNN to extract spatial features. The outputs are then passed into a recurrent sequence learning model (i.e. LSTM) to identify temporal features within the image sequence.

1. **What are the advantages of building an RNN using dynamic\_rnn() rather than static\_rnn()?**

Static

Internally, tf.nn.rnn creates an unrolled graph for a fixed RNN length. That means, if you call tf.nn.rnn with inputs having 200 time steps you are creating a static graph with 200 RNN steps. First, graph creation is slow. Second, you’re unable to pass in longer sequences (> 200) than you’ve originally specified.

Dynamic

tf.nn.dynamic\_rnn solves this. It uses a tf.While loop to dynamically construct the graph when it is executed. That means graph creation is faster and you can feed batches of variable size.

1. **How can you deal with variable-length input sequences? What about variable-length output sequences?**

The most commonly adopted solution is to truncate all inputs to the same length, which usually coincides with the shorter length input. However, this creates a huge loss of data, and as we know, data is gold to us.

One possible alternative is its opposite, which is padding (add data until all signals are at the same length). The problem with padding is that it adds data with no real sense, and also with very long inputs, the network becomes unsustainable in size. Of course, padding could be done via augmentation. However, in particular as regards the signals in which the order of the data is fundamental, applying augmentation is going to “dirty” this information.

1. **What is a common way to distribute training and execution of a deep RNN across multiple GPUs?**

Strategy is a TensorFlow API to distribute training across multiple GPUs