NLP assignments -4

1. A sequence-to-sequence RNN can be used for tasks such as machine translation, text summarization, and image caption generation. A sequence-to-vector RNN can be used for tasks such as sentiment analysis, where the input is a sequence (e.g. a sentence or a document) and the output is a single label (e.g. positive or negative). A vector-to-sequence RNN can be used for tasks such as music generation, where the input is a sequence of control parameters (e.g. chord progression) and the output is a sequence of audio samples.
2. Encoder-decoder RNNs are often used for machine translation because they allow the model to process the entire input sequence (encoder) and then generate the entire output sequence (decoder) rather than generating one output symbol at a time. This is useful for tasks such as machine translation because it allows the model to consider the context of the entire source sentence when generating the translation.
3. One way to combine a convolutional neural network with an RNN to classify videos would be to use the CNN to extract features from each frame of the video and feed the features into the RNN. The RNN would then process the sequence of features and make a prediction based on the entire sequence.
4. The advantage of using dynamic\_rnn() rather than static\_rnn() is that it allows the model to handle input sequences of variable length. With static\_rnn(), the input sequences must all have the same length, so any sequences that are shorter than the maximum length must be padded with zeros. This can be inefficient because it means that the model is processing a lot of unnecessary padding.
5. One way to deal with variable-length input sequences is to use padding. This involves adding a special symbol (e.g. a zero or a null token) to the end of each sequence until all the sequences have the same length. Another option is to use an RNN with attention mechanisms, which allow the model to focus on certain parts of the input sequence when processing it. To deal with variable-length output sequences, one option is to use a special end-of-sequence symbol and stop generating output when this symbol is encountered. Another option is to use a seq2seq model with a dynamic decoder, which allows the model to generate variable-length output sequences.
6. A common way to distribute the training and execution of a deep RNN across multiple GPUs is to use model parallelism, where different parts of the model (e.g. different layers) are trained on different GPUs. Another option is data parallelism, where different GPUs are used to process different parts of the data in parallel. Both model and data parallelism can be implemented using frameworks such as TensorFlow or PyTorch.

Top of Form