1.What are the pros and cons of using a stateful RNN versus a stateless RNN?

Ans.

Stateful RNNs can be more efficient as they reuse the hidden state between batches, making it possible to process longer sequences. However, they can also be more difficult to train and may require careful handling of the hidden state to avoid issues with exploding or vanishing gradients. Stateless RNNs, on the other hand, do not have these issues, but may require more memory and be less efficient when processing long sequences.

2.Why do people use Encoder–Decoder RNNs rather than plain sequence-to-sequence RNNs for automatic translation?

Encoder-Decoder RNNs are better suited for automatic translation as they can learn to represent the input sentence in a fixed-size vector, which can then be used to generate the output sentence. Plain sequence-to-sequence RNNs, on the other hand, generate the output sentence based on the last hidden state of the input sentence, which may not be sufficient to capture all the necessary information.

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

Ans.

One way to deal with variable-length input sequences is to use padding to make all the sequences the same length. Another approach is to use bucketing, which groups input sequences of similar lengths into the same batch. For variable-length output sequences, you can use techniques such as teacher forcing, which uses the ground-truth outputs as inputs during training, or beam search, which generates multiple output sequences and selects the one with the highest probability.

4.What is beam search and why would you use it? What tool can you use to implement it?

Ans.

Beam search is a search algorithm used to find the most likely output sequence in a sequence-to-sequence model. It generates multiple candidate sequences and keeps track of the top-k most probable sequences at each step, based on their accumulated probabilities. It is often used in tasks such as machine translation or speech recognition. It can be implemented using tools such as TensorFlow or PyTorch.

5.What is an attention mechanism? How does it help?

Ans.

An attention mechanism is a mechanism that allows a model to selectively focus on parts of the input sequence that are relevant to predicting the output at a given time step. It does this by assigning a weight to each input element, indicating its importance. The model can then use these weights to calculate a weighted sum of the input sequence, which is used to make a prediction. This can improve the performance of sequence-to-sequence models, particularly for long input sequences.

6.What is the most important layer in the Transformer architecture? What is its purpose?

Ans.

The most important layer in the Transformer architecture is the self-attention layer, which allows the model to selectively attend to different parts of the input sequence. It does this by calculating a set of attention scores between each pair of positions in the sequence, and then using these scores to calculate a weighted sum of the input sequence. This allows the model to capture long-range dependencies in the input sequence more effectively than RNN-based models.

7.When would you need to use sampled softmax?

Ans.

Sampled softmax is used when the output vocabulary is large, making it computationally expensive to calculate the probability of each possible output word. It randomly samples a subset of the output vocabulary and calculates the probabilities only for these words, which can make the training process faster and more efficient. It is often used in language modeling or machine translation tasks.