tf.contrib.legacy_seq2seq.embedding_rnn_seq2seq

```
embedding_rnn_seq2seq(
    encoder_inputs,
    decoder_inputs,
    cell,
    num_encoder_symbols,
    num_decoder_symbols,
    embedding_size,
    output_projection=None,
    feed_previous=False,
    dtype=None,
    scope=None
)
```

Defined in `tensorflow/contrib/legacy_seq2seq/python/ops/seq2seq.py`.

Embedding RNN sequence-to-sequence model.

This model first embeds encoder_inputs by a newly created embedding (of shape [num_encoder_symbols x input_size]). Then it runs an RNN to encode embedded encoder_inputs into a state vector. Next, it embeds decoder_inputs by another newly created embedding (of shape [num_decoder_symbols x input_size]). Then it runs RNN decoder, initialized with the last encoder state, on embedded decoder_inputs.

Args:

- `encoder_inputs`: A list of 1D int32 Tensors of shape [batch_size].
- `decoder_inputs`: A list of 1D int32 Tensors of shape [batch_size].
- `cell`: tf.nn.rnn_cell.RNNCell defining the cell function and size.
- `num_encoder_symbols`: Integer; number of symbols on the encoder side.
- `num_decoder_symbols`: Integer; number of symbols on the decoder side.
- `embedding_size`: Integer, the length of the embedding vector for each symbol.
- `output_projection`: None or a pair (W, B) of output projection weights and biases; W has shape [output_size x num_decoder_symbols] and B has shape [num_decoder_symbols]; if provided and feed_previous=True, each fed previous output will first be multiplied by W and added B.
- `feed_previous`: Boolean or scalar Boolean Tensor; if True, only the first of decoder_inputs will be used (the "GO" symbol), and all other decoder inputs will be taken from previous outputs (as in embedding_rnn_decoder). If False, decoder_inputs are used as given (the standard decoder case).
- `dtype`: The dtype of the initial state for both the encoder and encoder rnn cells (default: tf.float32).
- `scope`: VariableScope for the created subgraph; defaults to "embedding_rnn_seq2seq"

Returns:

A tuple of the form (outputs, state), where: `outputs`*: A list of the same length as decoder_inputs of 2D Tensors. The output is of shape [batch_size x cell.output_size] when output_projection is not None (and represents the dense representation of predicted tokens). It is of shape [batch_size x num_decoder_symbols] when output_projection is None.* `state`: The state of each decoder cell in each time-step. This is a list with length len(decoder_inputs) – one item for each time-step. It is a 2D

Tensor of shape [batch_size x cell.state_size].