tf.contrib.legacy_seq2seq.embedding_rnn_decoder

```
embedding_rnn_decoder(
    decoder_inputs,
    initial_state,
    cell,
    num_symbols,
    embedding_size,
    output_projection=None,
    feed_previous=False,
    update_embedding_for_previous=True,
    scope=None
)
```

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

RNN decoder with embedding and a pure-decoding option.

Args:

- `decoder_inputs`: A list of 1D batch-sized int32 Tensors (decoder inputs).
- `initial_state`: 2D Tensor [batch_size x cell.state_size].
- `cell`: tf.nn.rnn_cell.RNNCell defining the cell function.
- `num_symbols`: Integer, how many symbols come into the embedding.
- `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_symbols] and B has shape [num_symbols]; if provided and feed_previous=True, each fed previous output will first be multiplied by W and added B.
- `feed_previous`: Boolean; if True, only the first of decoder_inputs will be used (the "GO" symbol), and all other decoder inputs will be generated by: next = embedding_lookup(embedding, argmax(previous_output)), In effect, this implements a greedy decoder. It can also be used during training to emulate http://arxiv.org/abs/1506.03099. If False, decoder_inputs are used as given (the standard decoder case).
- `update_embedding_for_previous`: Boolean; if False and feed_previous=True, only the embedding for the first symbol of decoder_inputs (the "GO" symbol) will be updated by back propagation. Embeddings for the symbols generated from the decoder itself remain unchanged. This parameter has no effect if feed_previous=False.
- `scope`: VariableScope for the created subgraph; defaults to "embedding_rnn_decoder".

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].

Raises:

- `ValueError`: When output_projection has the wrong shape.

Last updated November 2, 2017.