tf.contrib.rnn.stack_bidirectional_dynamic_rnn

```
stack_bidirectional_dynamic_rnn(
    cells_fw,
    cells_bw,
    inputs,
    initial_states_fw=None,
    initial_states_bw=None,
    dtype=None,
    sequence_length=None,
    parallel_iterations=None,
    time_major=False,
    scope=None
)
```

Defined in `tensorflow/contrib/rnn/python/ops/rnn.py`.

See the guide: RNN and Cells (contrib) > Recurrent Neural Networks

Creates a dynamic bidirectional recurrent neural network.

Stacks several bidirectional rnn layers. The combined forward and backward layer outputs are used as input of the next layer. tf.bidirectional_rnn does not allow to share forward and backward information between layers. The input_size of the first forward and backward cells must match. The initial state for both directions is zero and no intermediate states are returned.

Args:

- `cells_fw`: List of instances of RNNCell, one per layer, to be used for forward direction.
- `cells_bw`: List of instances of RNNCell, one per layer, to be used for backward direction.
- `inputs`: The RNN inputs. this must be a tensor of shape: `[batch_size, max_time, ...]`, or a nested tuple of such elements.
- `initial_states_fw`: (optional) A list of the initial states (one per layer) for the forward RNN. Each tensor must has an appropriate type and shape `[batch_size, cell_fw.state_size]`.
- `initial_states_bw`: (optional) Same as for `initial_states_fw`, but using the corresponding properties of `cells_bw`.
- `dtype`: (optional) The data type for the initial state. Required if either of the initial states are not provided.
- `sequence_length`: (optional) An int32/int64 vector, size `[batch_size]`, containing the actual lengths for each of the sequences.
- `parallel_iterations`: (Default: 32). The number of iterations to run in parallel. Those operations which do not have any temporal dependency and can be run in parallel, will be. This parameter trades off time for space. Values >> 1 use more memory but take less time, while smaller values use less memory but computations take longer.
- `time_major`: The shape format of the inputs and outputs Tensors. If true, these Tensors must be shaped [max_time, batch_size, depth]. If false, these Tensors must be shaped [batch_size, max_time, depth]. Using time_major = True is a bit more efficient because it avoids transposes at the beginning and end of the RNN calculation. However, most TensorFlow data is batch-major, so by default this function accepts input and emits output in batch-major form.
- `scope`: VariableScope for the created subgraph; defaults to None.

Returns:

A tuple (outputs, output_state_fw, output_state_bw) where: * `outputs`: Output `Tensor` shaped: `batch_size, max_time, layers_output]`. Where layers_output are depth-concatenated forward and backward outputs. output_states_fw is the final states, one tensor per layer, of the forward rnn. output_states_bw is the final states, one tensor per layer, of the backward rnn.

Raises:

- `TypeError`: If `cell_fw` or `cell_bw` is not an instance of `RNNCell`.
- `ValueError`: If inputs is `None`.
