tf.contrib.seq2seq.BahdanauAttention

Class **`BahdanauAttention`**

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

See the guide: Seq2seq Library (contrib) > Attention

Implements Bahdanau-style (additive) attention.

This attention has two forms. The first is Bahdanau attention, as described in:

Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate." ICLR 2015. https://arxiv.org/abs/1409.0473

The second is the normalized form. This form is inspired by the weight normalization article:

Tim Salimans, Diederik P. Kingma. "Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks." https://arxiv.org/abs/1602.07868

To enable the second form, construct the object with parameter `normalize=True`.

Properties

`alignments_size`

`batch_size`

`keys`

`memory_layer`

`query_layer`

`values`

Methods

`__init__`

```
__init__(
    num_units,
    memory,
    memory_sequence_length=None,
    normalize=False,
    probability_fn=None,
    score_mask_value=float('-inf'),
    name='BahdanauAttention'
)
```

Construct the Attention mechanism.

Args:

- `num_units`: The depth of the query mechanism.
- `memory`: The memory to query; usually the output of an RNN encoder. This tensor should be shaped `[batch_size, max_time, ...]`. memory_sequence_length (optional): Sequence lengths for the batch entries in memory. If provided, the memory tensor rows are masked with zeros for values past the respective sequence lengths.
- `normalize`: Python boolean. Whether to normalize the energy term.
- `probability_fn`: (optional) A `callable`. Converts the score to probabilities. The default is `tf.nn.softmax`. Other options include `tf.contrib.seq2seq.hardmax` and `tf.contrib.sparsemax.sparsemax`. Its signature should be: `probabilities = probability_fn(score)`.
- `score_mask_value`: (optional): The mask value for score before passing into `probability_fn`. The default is -inf. Only used if `memory_sequence_length` is not None.
- `name`: Name to use when creating ops.

`__call__`

```
__call__(
    query,
    previous_alignments
)
```

Score the query based on the keys and values.

Args:

- `query`: Tensor of dtype matching `self.values` and shape `[batch_size, query_depth]`.
- `previous_alignments`: Tensor of dtype matching `self.values` and shape `[batch_size, alignments_size]` (`alignments_size` is memory's `max_time`).

Returns:

- `alignments`: Tensor of dtype matching `self.values` and shape `[batch_size, alignments_size]` (`alignments_size` is memory's `max_time`).

`initial_alignments`

```
initial_alignments(
    batch_size,
    dtype
)
```

Creates the initial alignment values for the `AttentionWrapper` class.

This is important for AttentionMechanisms that use the previous alignment to calculate the alignment at the next time step (e.g. monotonic attention).

The default behavior is to return a tensor of all zeros.

Args:

- `batch_size`: `int32` scalar, the batch_size.
- `dtype`: The `dtype`.

Returns:

A `dtype` tensor shaped `[batch_size, alignments_size]` (`alignments_size` is the values' `max_time`).