tf.sparse_merge

```
sparse_merge(
    sp_ids,
    sp_values,
    vocab_size,
    name=None,
    already_sorted=False
)
```

Defined in `tensorflow/python/ops/sparse_ops.py`.

See the guide: Sparse Tensors > Conversion

Combines a batch of feature ids and values into a single `SparseTensor`.

The most common use case for this function occurs when feature ids and their corresponding values are stored in `Example` protos on disk. `parse_example` will return a batch of ids and a batch of values, and this function joins them into a single logical `SparseTensor` for use in functions such as `sparse_tensor_dense_matmul`, `sparse_to_dense`, etc.

The `SparseTensor` returned by this function has the following properties:

- `indices` is equivalent to `sp_ids.indices` with the last dimension discarded and replaced with `sp_ids.values`.
- `values` is simply `sp_values.values`.
- If `sp_ids.dense_shape = [D0, D1, ..., Dn, K]`, then `output.shape = [D0, D1, ..., Dn, vocab_size]`.

For example, consider the following feature vectors:

```
  vector1 = [-3, 0, 0, 0, 0, 0]
  vector2 = [ 0, 1, 0, 4, 1, 0]
  vector3 = [ 5, 0, 0, 9, 0, 0]
```

These might be stored sparsely in the following Example protos by storing only the feature ids (column number if the vectors are treated as a matrix) of the non-zero elements and the corresponding values:

```
  examples = [Example(features={
                  "ids": Feature(int64_list=Int64List(value=[0])),
                  "values": Feature(float_list=FloatList(value=[-3]))}),
              Example(features={
                  "ids": Feature(int64_list=Int64List(value=[1, 4, 3])),
                  "values": Feature(float_list=FloatList(value=[1, 1, 4]))}),
              Example(features={
                  "ids": Feature(int64_list=Int64List(value=[0, 3])),
                  "values": Feature(float_list=FloatList(value=[5, 9]))})]
```

The result of calling parse_example on these examples will produce a dictionary with entries for "ids" and "values". Passing those two objects to this function along with vocab_size=6, will produce a `SparseTensor` that sparsely represents all three instances. Namely, the `indices` property will contain the coordinates of the non-zero entries in the feature matrix (the first dimension is the row number in the matrix, i.e., the index within the batch, and the second dimension is the column number, i.e., the feature id); `values` will contain the actual values. `shape` will be the shape of the original matrix, i.e., (3, 6). For our example above, the output will be equal to:

```
SparseTensor(indices=[[0, 0], [1, 1], [1, 3], [1, 4], [2, 0], [2, 3]],
             values=[-3, 1, 4, 1, 5, 9],
             dense_shape=[3, 6])
```

This method generalizes to higher-dimensions by simply providing a list for both the sp_ids as well as the vocab_size. In this case the resulting `SparseTensor` has the following properties: - `indices` is equivalent to `sp_ids[0].indices` with the last dimension discarded and concatenated with `sp_ids[0].values, sp_ids[1].values, ...`. - `values` is simply `sp_values.values`. - If `sp_ids.dense_shape = [D0, D1, ..., Dn, K]`, then `output.shape = [D0, D1, ..., Dn] + vocab_size`.

Args:

- `sp_ids`: A single `SparseTensor` with `values` property of type `int32` or `int64` or a Python list of such `SparseTensor`s or a list thereof.
- `sp_values`: A `SparseTensor` of any type.
- `vocab_size`: A scalar `int64` Tensor (or Python int) containing the new size of the last dimension, `all(0 <= sp_ids.values < vocab_size)`. Or a list thereof with `all(0 <= sp_ids[i].values < vocab_size[i])` for all `i`.
- `name`: A name prefix for the returned tensors (optional)
- `already_sorted`: A boolean to specify whether the per-batch values in `sp_values` are already sorted. If so skip sorting, False by default (optional).

Returns:

A `SparseTensor` compactly representing a batch of feature ids and values, useful for passing to functions that expect such a `SparseTensor`.

Raises:

- `TypeError`: If `sp_values` is not a `SparseTensor`. Or if `sp_ids` is neither a `SparseTensor` nor a list thereof. Or if `vocab_size` is not a `Tensor` or a Python int and `sp_ids` is a `SparseTensor`. Or if `vocab_size` is not a or list thereof and `sp_ids` is a list.
- `ValueError`: If `sp_ids` and `vocab_size` are lists of different lengths.

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License, and code samples are licensed under the Apache 2.0 License. For details, see our Site Policies. Java is a registered trademark of Oracle and/or its affiliates.

Last updated November 2, 2017.

English

Terms | Privacy