tf.contrib.lookup.MutableHashTable

Contents

Class `MutableHashTable`

Inherits From: `LookupInterface`

Defined in `tensorflow/contrib/lookup/lookup_ops.py`.

A generic mutable hash table implementation.

Data can be inserted by calling the insert method. It does not support initialization via the init method.

Example usage:

```
table = tf.contrib.lookup.MutableHashTable(key_dtype=tf.string,
                                           value_dtype=tf.int64,
                                           default_value=-1)
table.insert(keys, values)
out = table.lookup(query_keys)
print(out.eval())
```

Properties

`init`

The table initialization op.

`key_dtype`

The table key dtype.

`name`

The name of the table.

`value_dtype`

The table value dtype.

Methods

`__init__`

```
__init__(
    key_dtype,
    value_dtype,
    default_value,
    shared_name=None,
    name='MutableHashTable',
    checkpoint=True
)
```

Creates an empty `MutableHashTable` object.

Creates a table, the type of its keys and values are specified by key_dtype and value_dtype, respectively.

Args:

- `key_dtype`: the type of the key tensors.
- `value_dtype`: the type of the value tensors.
- `default_value`: The value to use if a key is missing in the table.
- `shared_name`: If non-empty, this table will be shared under the given name across multiple sessions.
- `name`: A name for the operation (optional).
- `checkpoint`: if True, the contents of the table are saved to and restored from checkpoints. If `shared_name` is empty for a checkpointed table, it is shared using the table node name.

Returns:

A `MutableHashTable` object.

Raises:

- `ValueError`: If checkpoint is True and no name was specified.

`export`

```
export(name=None)
```

Returns tensors of all keys and values in the table.

Args:

- `name`: A name for the operation (optional).

Returns:

A pair of tensors with the first tensor containing all keys and the second tensors containing all values in the table.

`insert`

```
insert(
    keys,
    values,
    name=None
)
```

Associates `keys` with `values`.

Args:

- `keys`: Keys to insert. Can be a tensor of any shape. Must match the table's key type.
- `values`: Values to be associated with keys. Must be a tensor of the same shape as `keys` and match the table's value type.
- `name`: A name for the operation (optional).

Returns:

The created Operation.

Raises:

- `TypeError`: when `keys` or `values` doesn't match the table data types.

`lookup`

```
lookup(
    keys,
    name=None
)
```

Looks up `keys` in a table, outputs the corresponding values.

The `default_value` is used for keys not present in the table.

Args:

- `keys`: Keys to look up. Can be a tensor of any shape. Must match the table's key_dtype.
- `name`: A name for the operation (optional).

Returns:

A tensor containing the values in the same shape as `keys` using the table's value type.

Raises:

- `TypeError`: when `keys` do not match the table data types.

`size`

```
size(name=None)
```

Compute the number of elements in this table.

Args:

- `name`: A name for the operation (optional).

Returns:

A scalar tensor containing the number of elements in this table.

Last updated November 2, 2017.