tf.contrib.lookup.IdTableWithHashBuckets

Contents

Class `IdTableWithHashBuckets`

Inherits From: `LookupInterface`

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

String to Id table wrapper that assigns out-of-vocabulary keys to buckets.

For example, if an instance of `IdTableWithHashBuckets` is initialized with a string-to-id table that maps: - emerson -> 0 - lake -> 1 - palmer -> 2

The `IdTableWithHashBuckets` object will performs the following mapping: - emerson -> 0 - lake -> 1 - palmer -> 2 - -> bucket id between 3 and 3 + num_oov_buckets - 1, calculated by: hash() % num_oov_buckets + vocab_size

If input_tensor is ["emerson", "lake", "palmer", "king", "crimson"], the lookup result is [0, 1, 2, 4, 7]

If `table` is None, only out-of-vocabulary buckets are used.

Example usage:

```
num_oov_buckets = 3
input_tensor = tf.constant(["emerson", "lake", "palmer", "king", "crimnson"])
table = tf.IdTableWithHashBuckets(
    tf.HashTable(tf.TextFileIdTableInitializer(filename), default_value),
    num_oov_buckets)
out = table.lookup(input_tensor).
table.init.run()
print(out.eval())
```

The hash function used for generating out-of-vocabulary buckets ID is handled by `hasher_spec`.

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__(
    table,
    num_oov_buckets,
    hasher_spec=tf.contrib.lookup.FastHashSpec,
    name=None,
    key_dtype=None
)
```

Construct a `IdTableWithHashBuckets` object.

Args:

- `table`: Table that maps `tf.string` or `tf.int64` keys to `tf.int64` ids.
- `num_oov_buckets`: Number of buckets to use for out-of-vocabulary keys.
- `hasher_spec`: A `HasherSpec` to specify the hash function to use for assignation of out-of-vocabulary buckets (optional).
- `name`: A name for the operation (optional).
- `key_dtype`: Data type of keys passed to `lookup`. Defaults to `table.key_dtype` if `table` is specified, otherwise `tf.string`. Must be string or integer, and must be castable to `table.key_dtype`.

Raises:

- `ValueError`: when `table` in None and `num_oov_buckets` is not positive.
- `TypeError`: when `hasher_spec` is invalid.

`lookup`

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

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

It assigns out-of-vocabulary keys to buckets based in their hashes.

Args:

- `keys`: Keys to look up. May be either a `SparseTensor` or dense `Tensor`.

- `name`: Optional name for the op.

Returns:

A `SparseTensor` if keys are sparse, otherwise a dense `Tensor`.

Raises:

- `TypeError`: when `keys` doesn't match the table key data type.

`size`

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

Compute the number of elements in this table.
