Module: tf.lookup

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup#top_of_page)
* [Modules](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup#modules)
* [Classes](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup#classes)

Public API for tf.lookup namespace.

Modules

[experimental](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/experimental) module: Public API for tf.lookup.experimental namespace.

Classes

[class KeyValueTensorInitializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/KeyValueTensorInitializer): Table initializers given keys and values tensors.

[class StaticHashTable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticHashTable): A generic hash table implementation.

[class StaticVocabularyTable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticVocabularyTable): String to Id table wrapper that assigns out-of-vocabulary keys to buckets.

[class TextFileIndex](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/TextFileIndex): The key and value content to get from each line.

[class TextFileInitializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/TextFileInitializer): Table initializers from a text file.

# tf.lookup.KeyValueTensorInitializer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/KeyValueTensorInitializer#top_of_page)
* [Class KeyValueTensorInitializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/KeyValueTensorInitializer#class_keyvaluetensorinitializer)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/KeyValueTensorInitializer#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/KeyValueTensorInitializer#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/KeyValueTensorInitializer#properties)

## Class KeyValueTensorInitializer

Table initializers given keys and values tensors.

### Aliases:

* Class tf.compat.v1.lookup.KeyValueTensorInitializer
* Class tf.compat.v2.lookup.KeyValueTensorInitializer
* Class tf.lookup.KeyValueTensorInitializer

Defined in [python/ops/lookup\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/lookup_ops.py).

## \_\_init\_\_

\_\_init\_\_(  
    keys,  
    values,  
    key\_dtype=None,  
    value\_dtype=None,  
    name=None  
)

Constructs a table initializer object based on keys and values tensors.

#### Args:

* **keys**: The tensor for the keys.
* **values**: The tensor for the values.
* **key\_dtype**: The keys data type. Used when keys is a python array.
* **value\_dtype**: The values data type. Used when values is a python array.
* **name**: A name for the operation (optional).

## Properties

### key\_dtype

The expected table key dtype.

**value\_dtype**

The expected table value dtype.

## Methods

## initialize

initialize(table)

Initializes the given table with keys and values tensors.

#### Args:

* **table**: The table to initialize.

#### Returns:

The operation that initializes the table.

#### Raises:

* **TypeError**: when the keys and values data types do not match the table key and value data types.

# tf.lookup.StaticHashTable

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticHashTable#top_of_page)
* [Class StaticHashTable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticHashTable#class_statichashtable)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticHashTable#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticHashTable#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticHashTable#properties)

## Class StaticHashTable

A generic hash table implementation.

### Aliases:

* Class tf.compat.v2.lookup.StaticHashTable
* Class tf.lookup.StaticHashTable

Defined in [python/ops/lookup\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/lookup_ops.py).

#### Example usage:

table = tf.lookup.StaticHashTable(  
    tf.KeyValueTensorInitializer(keys, values), -1)  
out = table.lookup(input\_tensor)  
table.init.run()  
print(out.eval())

## \_\_init\_\_

\_\_init\_\_(  
    initializer,  
    default\_value,  
    name=None  
)

Creates a non-initialized HashTable object.

Creates a table, the type of its keys and values are specified by the initializer. Before using the table you will have to initialize it. After initialization the table will be immutable.

#### Args:

* **initializer**: The table initializer to use. See HashTable kernel for supported key and value types.
* **default\_value**: The value to use if a key is missing in the table.
* **name**: A name for the operation (optional).

#### Returns:

A HashTable object.

## Properties

### default\_value

The default value of the table.

### key\_dtype

The table key dtype.

**Name**

**resource\_handle**

Returns the resource handle associated with this Resource.

**value\_dtype**

The table value dtype.

## Methods

## 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.

### 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. May be either a SparseTensor or dense Tensor.
* **name**: A name for the operation (optional).

#### Returns:

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

#### Raises:

* **TypeError**: when keys or default\_value doesn't 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.

# tf.lookup.StaticVocabularyTable

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticVocabularyTable#top_of_page)
* [Class StaticVocabularyTable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticVocabularyTable#class_staticvocabularytable)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticVocabularyTable#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticVocabularyTable#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/StaticVocabularyTable#properties)

## Class StaticVocabularyTable

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

### Aliases:

* Class tf.compat.v2.lookup.StaticVocabularyTable
* Class tf.lookup.StaticVocabularyTable

Defined in [python/ops/lookup\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/lookup_ops.py).

For example, if an instance of StaticVocabularyTable is initialized with a string-to-id initializer that maps:

* emerson -> 0
* lake -> 1
* palmer -> 2

The Vocabulary object will performs the following mapping:

* emerson -> 0
* lake -> 1
* palmer -> 2
* <other term> -> bucket\_id, where bucket\_id will be between 3 and 3 + num\_oov\_buckets - 1, calculated by: hash(<term>) % num\_oov\_buckets + vocab\_size

If input\_tensor is ["emerson", "lake", "palmer", "king", "crimson"], the lookup result is [0, 1, 2, 4, 7].

If initializer 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.lookup.StaticVocabularyTable(  
    tf.TextFileIdTableInitializer(filename), 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 Fingerprint64.

## \_\_init\_\_

\_\_init\_\_(  
    initializer,  
    num\_oov\_buckets,  
    lookup\_key\_dtype=None,  
    name=None  
)

Construct a StaticVocabularyTable object.

#### Args:

* **initializer**: A TableInitializerBase object that contains the data used to initialize the table. If None, then we only use out-of-vocab buckets.
* **num\_oov\_buckets**: Number of buckets to use for out-of-vocabulary keys. Must be greater than zero.
* **lookup\_key\_dtype**: Data type of keys passed to lookup. Defaults toinitializer.key\_dtype if initializer is specified, otherwise [tf.string](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf#string). Must be string or integer, and must be castable to initializer.key\_dtype.
* **name**: A name for the operation (optional).

#### Raises:

* **ValueError**: when num\_oov\_buckets is not positive.
* **TypeError**: when lookup\_key\_dtype or initializer.key\_dtype are not integer or string. Also when initializer.value\_dtype != int64.

## Properties

### key\_dtype

The table key dtype.

### name

### resource\_handle

### value\_dtype

The table value dtype.

## Methods

### 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.

# tf.lookup.TextFileIndex

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/TextFileIndex#top_of_page)
* [Class TextFileIndex](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/TextFileIndex#class_textfileindex)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/TextFileIndex#aliases)
* [Class Members](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/TextFileIndex#class_members)

## Class TextFileIndex

The key and value content to get from each line.

### Aliases:

* Class tf.compat.v1.lookup.TextFileIndex
* Class tf.compat.v2.lookup.TextFileIndex
* Class tf.lookup.TextFileIndex

Defined in [python/ops/lookup\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/lookup_ops.py).

This class defines the key and value used for tf.lookup.TextFileInitializer.

The key and value content to get from each line is specified either by the following, or a value >=0. \* TextFileIndex.LINE\_NUMBER means use the line number starting from zero, expects data type int64. \* TextFileIndex.WHOLE\_LINE means use the whole line content, expects data type string.

A value >=0 means use the index (starting at zero) of the split line based on delimiter.

## Class Members

* LINE\_NUMBER = -1
* WHOLE\_LINE = -2

# tf.lookup.TextFileInitializer

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/TextFileInitializer#top_of_page)
* [Class TextFileInitializer](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/TextFileInitializer#class_textfileinitializer)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/TextFileInitializer#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/TextFileInitializer#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/TextFileInitializer#properties)

## Class TextFileInitializer

Table initializers from a text file.

### Aliases:

* Class tf.compat.v1.lookup.TextFileInitializer
* Class tf.compat.v2.lookup.TextFileInitializer
* Class tf.lookup.TextFileInitializer

Defined in [python/ops/lookup\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/lookup_ops.py).

This initializer assigns one entry in the table for each line in the file.

The key and value type of the table to initialize is given by key\_dtype and value\_dtype.

The key and value content to get from each line is specified by the key\_index and value\_index.

* TextFileIndex.LINE\_NUMBER means use the line number starting from zero, expects data type int64.
* TextFileIndex.WHOLE\_LINE means use the whole line content, expects data type string.
* A value >=0 means use the index (starting at zero) of the split line based on delimiter.

For example if we have a file with the following content:

emerson 10  
lake 20  
palmer 30

The following snippet initializes a table with the first column as keys and second column as values:

* emerson -> 10
* lake -> 20
* palmer -> 30

table = tf.lookup.StaticHashTable(tf.lookup.TextFileInitializer(  
    "test.txt", tf.string, 0, tf.int64, 1, delimiter=" "), -1)  
...  
table.init.run()

Similarly to initialize the whole line as keys and the line number as values.

* emerson 10 -> 0
* lake 20 -> 1
* palmer 30 -> 2

table = tf.lookup.StaticHashTable(tf.lookup.TextFileInitializer(  
    "test.txt", tf.string, tf.lookup.TextFileIndex.WHOLE\_LINE,  
    tf.int64, tf.lookup.TextFileIndex.LINE\_NUMBER, delimiter=" "), -1)  
...  
table.init.run()

## \_\_init\_\_

\_\_init\_\_(  
    filename,  
    key\_dtype,  
    key\_index,  
    value\_dtype,  
    value\_index,  
    vocab\_size=None,  
    delimiter='\t',  
    name=None  
)

Constructs a table initializer object to populate from a text file.

It generates one key-value pair per line. The type of table key and value are specified by key\_dtypeand value\_dtype, respectively. Similarly the content of the key and value are specified by the key\_index and value\_index.

* TextFileIndex.LINE\_NUMBER means use the line number starting from zero, expects data type int64.
* TextFileIndex.WHOLE\_LINE means use the whole line content, expects data type string.
* A value >=0 means use the index (starting at zero) of the split line based on delimiter.

#### Args:

* **filename**: The filename of the text file to be used for initialization. The path must be accessible from wherever the graph is initialized (eg. trainer or eval workers). The filename may be a scalar Tensor.
* **key\_dtype**: The key data type.
* **key\_index**: the index that represents information of a line to get the table 'key' values from.
* **value\_dtype**: The value data type.
* **value\_index**: the index that represents information of a line to get the table 'value' values from.'
* **vocab\_size**: The number of elements in the file, if known.
* **delimiter**: The delimiter to separate fields in a line.
* **name**: A name for the operation (optional).

#### Raises:

* **ValueError**: when the filename is empty, or when the table key and value data types do not match the expected data types.

## Properties

### key\_dtype

The expected table key dtype.

### value\_dtype

The expected table value dtype.

## Methods

### initialize

initialize(table)

Initializes the table from a text file.

#### Args:

* **table**: The table to be initialized.

#### Returns:

The operation that initializes the table.

#### Raises:

* **TypeError**: when the keys and values data types do not match the table key and value data types.

# tf.lookup.experimental.DenseHashTable

* [**Contents**](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/experimental/DenseHashTable#top_of_page)
* [Class DenseHashTable](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/experimental/DenseHashTable#class_densehashtable)
  + [Aliases:](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/experimental/DenseHashTable#aliases)
* [\_\_init\_\_](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/experimental/DenseHashTable#__init__)
* [Properties](https://www.tensorflow.org/versions/r2.0/api_docs/python/tf/lookup/experimental/DenseHashTable#properties)

## Class DenseHashTable

A generic mutable hash table implementation using tensors as backing store.

### Aliases:

* Class tf.compat.v1.lookup.experimental.DenseHashTable
* Class tf.compat.v2.lookup.experimental.DenseHashTable
* Class tf.lookup.experimental.DenseHashTable

Defined in [python/ops/lookup\_ops.py](https://github.com/tensorflow/tensorflow/tree/r2.0/tensorflow/python/ops/lookup_ops.py).

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

It uses "open addressing" with quadratic reprobing to resolve collisions. Compared to MutableHashTable the insert, remove and lookup operations in a DenseHashTable are typically faster, but memory usage can be higher. However, DenseHashTable does not require additional memory for temporary tensors created during checkpointing and restore operations.

#### Example usage:

table = tf.lookup.DenseHashTable(key\_dtype=tf.int64,  
                                 value\_dtype=tf.int64,  
                                 default\_value=-1,  
                                 empty\_key=0,  
                                 deleted\_key=-1)  
  
sess.run(table.insert(keys, values))  
out = table.lookup(query\_keys)  
print(out.eval())

## \_\_init\_\_

\_\_init\_\_(  
    key\_dtype,  
    value\_dtype,  
    default\_value,  
    empty\_key,  
    deleted\_key,  
    initial\_num\_buckets=None,  
    name='MutableDenseHashTable',  
    checkpoint=True  
)

Creates an empty DenseHashTable 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.
* **empty\_key**: the key to use to represent empty buckets internally. Must not be used in insert, remove or lookup operations.
* **deleted\_key**: the key to use to represent deleted buckets internally. Must not be used in insert, remove or lookup operations and be different from the empty\_key.
* **initial\_num\_buckets**: the initial number of buckets.
* **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 DenseHashTable object.

#### Raises:

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

## Properties

### key\_dtype

The table key dtype.

### name

### resource\_handle

Returns the resource handle associated with this Resource.

### value\_dtype

The table value dtype.

## Methods

## erase

erase(  
    keys,  
    name=None  
)

Removes keys and its associated values from the table.

If a key is not present in the table, it is silently ignored.

#### Args:

* **keys**: Keys to remove. Can be a tensor of any shape. Must match the table's key type.
* **name**: A name for the operation (optional).

#### Returns:

The created Operation.

#### Raises:

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

### 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.
* insert\_or\_assign

insert\_or\_assign(  
    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.

### remove

remove(  
    keys,  
    name=None  
)

Removes keys and its associated values from the table.

If a key is not present in the table, it is silently ignored.

#### Args:

* **keys**: Keys to remove. Can be a tensor of any shape. Must match the table's key type.
* **name**: A name for the operation (optional).

#### Returns:

The created Operation.

#### 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.