tf.feature_column.crossed_column

```
crossed_column(
    keys,
    hash_bucket_size,
    hash_key=None
)
```

Defined in `tensorflow/python/feature_column/feature_column.py`.

Returns a column for performing crosses of categorical features.

Crossed features will be hashed according to `hash_bucket_size`. Conceptually, the transformation can be thought of as: Hash(cartesian product of features) % `hash_bucket_size`

For example, if the input features are:

- SparseTensor referred by first key:

```
shape = [2, 2]
{
    [0, 0]: "a"
    [1, 0]: "b"
    [1, 1]: "c"
}
```

- SparseTensor referred by second key:

```
shape = [2, 1]
{
    [0, 0]: "d"
    [1, 0]: "e"
}
```

then crossed feature will look like:

```
shape = [2, 2]
{
    [0, 0]: Hash64("d", Hash64("a")) % hash_bucket_size
    [1, 0]: Hash64("e", Hash64("b")) % hash_bucket_size
    [1, 1]: Hash64("e", Hash64("c")) % hash_bucket_size
}
```

Here is an example to create a linear model with crosses of string features:

```
keywords_x_doc_terms = crossed_column(['keywords', 'doc_terms'], 50K)
columns = [keywords_x_doc_terms, ...]
features = tf.parse_example(..., features=make_parse_example_spec(columns))
linear_prediction = linear_model(features, columns)
```

You could also use vocabulary lookup before crossing:

```
keywords = categorical_column_with_vocabulary_file(
    'keywords', '/path/to/vocabulary/file', vocabulary_size=1K)
keywords_x_doc_terms = crossed_column([keywords, 'doc_terms'], 50K)
columns = [keywords_x_doc_terms, ...]
features = tf.parse_example(..., features=make_parse_example_spec(columns))
linear_prediction = linear_model(features, columns)
```

If an input feature is of numeric type, you can use `categorical_column_with_identity`, or `bucketized_column`, as in the example:

```
# vertical_id is an integer categorical feature.
vertical_id = categorical_column_with_identity('vertical_id', 10K)
price = numeric_column('price')
# bucketized_column converts numerical feature to a categorical one.
bucketized_price = bucketized_column(price, boundaries=[...])
vertical_id_x_price = crossed_column([vertical_id, bucketized_price], 50K)
columns = [vertical_id_x_price, ...]
features = tf.parse_example(..., features=make_parse_example_spec(columns))
linear_prediction = linear_model(features, columns)
```

To use crossed column in DNN model, you need to add it in an embedding column as in this example:

```
vertical_id_x_price = crossed_column([vertical_id, bucketized_price], 50K)
vertical_id_x_price_embedded = embedding_column(vertical_id_x_price, 10)
dense_tensor = input_layer(features, [vertical_id_x_price_embedded, ...])
```

Args:

- `keys`: An iterable identifying the features to be crossed. Each element can be either:
  - string: Will use the corresponding feature which must be of string type.
  - `_CategoricalColumn`: Will use the transformed tensor produced by this column. Does not support hashed categorical column.
- `hash_bucket_size`: An int > 1. The number of buckets.
- `hash_key`: Specify the hash_key that will be used by the `FingerprintCat64` function to combine the crosses fingerprints on SparseCrossOp (optional).

Returns:

A `_CrossedColumn`.

Raises:

- `ValueError`: If `len(keys) < 2`.
- `ValueError`: If any of the keys is neither a string nor `_CategoricalColumn`.
- `ValueError`: If any of the keys is `_HashedCategoricalColumn`.
- `ValueError`: If `hash_bucket_size < 1`.

Stay Connected

Blog

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow

English

Terms | **Privacy**