tf.feature_column.categorical_column_with_hash_bucket

```
categorical_column_with_hash_bucket(
    key,
    hash_bucket_size,
    dtype=tf.string
)
```

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

Represents sparse feature where ids are set by hashing.

Use this when your sparse features are in string or integer format, and you want to distribute your inputs into a finite number of buckets by hashing. output_id = Hash(input_feature_string) % bucket_size

For input dictionary `features`, `features[key]` is either `Tensor` or `SparseTensor`. If `Tensor`, missing values can be represented by `-1` for int and `''` for string. Note that these values are independent of the `default_value` argument.

Example:

```
keywords = categorical_column_with_hash_bucket("keywords", 10K)
columns = [keywords, ...]
features = tf.parse_example(..., features=make_parse_example_spec(columns))
linear_prediction = linear_model(features, columns)

# or
keywords_embedded = embedding_column(keywords, 16)
columns = [keywords_embedded, ...]
features = tf.parse_example(..., features=make_parse_example_spec(columns))
dense_tensor = input_layer(features, columns)
```

Args:

- `key`: A unique string identifying the input feature. It is used as the column name and the dictionary key for feature parsing configs, feature `Tensor` objects, and feature columns.
- `hash_bucket_size`: An int > 1. The number of buckets.
- `dtype`: The type of features. Only string and integer types are supported.

Returns:

A `_HashedCategoricalColumn`.

Raises:

- `ValueError`: `hash_bucket_size` is not greater than 1.
- `ValueError`: `dtype` is neither string nor integer.

Last updated November 2, 2017.