tf.feature_column.categorical_column_with_vocabulary_list

```
categorical_column_with_vocabulary_list(
    key,
    vocabulary_list,
    dtype=None,
    default_value=-1,
    num_oov_buckets=0
)
```

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

A `_CategoricalColumn` with in-memory vocabulary.

Use this when your inputs are in string or integer format, and you have an in-memory vocabulary mapping each value to an integer ID. By default, out-of-vocabulary values are ignored. Use either (but not both) of `num_oov_buckets` and `default_value` to specify how to include out-of-vocabulary values.

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 with `num_oov_buckets`: In the following example, each input in `vocabulary_list` is assigned an ID 0-3 corresponding to its index (e.g., input 'B' produces output 2). All other inputs are hashed and assigned an ID 4-5.

```
colors = categorical_column_with_vocabulary_list(
    key='colors', vocabulary_list=('R', 'G', 'B', 'Y'),
    num_oov_buckets=2)
columns = [colors, ...]
features = tf.parse_example(..., features=make_parse_example_spec(columns))
linear_prediction, _, _ = linear_model(features, columns)
```

Example with `default_value`: In the following example, each input in `vocabulary_list` is assigned an ID 0-4 corresponding to its index (e.g., input 'B' produces output 3). All other inputs are assigned `default_value` 0.

```
colors = categorical_column_with_vocabulary_list(
    key='colors', vocabulary_list=('X', 'R', 'G', 'B', 'Y'), default_value=0)
columns = [colors, ...]
features = tf.parse_example(..., features=make_parse_example_spec(columns))
linear_prediction, _, _ = linear_model(features, columns)
```

And to make an embedding with either:

```
columns = [embedding_column(colors, 3),...]
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.
- `vocabulary_list`: An ordered iterable defining the vocabulary. Each feature is mapped to the index of its value (if present) in `vocabulary_list`. Must be castable to `dtype`.

- `dtype`: The type of features. Only string and integer types are supported. If `None`, it will be inferred from `vocabulary_list`.
- `default_value`: The integer ID value to return for out-of-vocabulary feature values, defaults to `-1`. This can not be specified with a positive `num_oov_buckets`.
- `num_oov_buckets`: Non-negative integer, the number of out-of-vocabulary buckets. All out-of-vocabulary inputs will be assigned IDs in the range `[len(vocabulary_list), len(vocabulary_list)+num_oov_buckets)` based on a hash of the input value. A positive `num_oov_buckets` can not be specified with `default_value`.

Returns:

A `_CategoricalColumn` with in-memory vocabulary.

Raises:

- `ValueError`: if `vocabulary_list` is empty, or contains duplicate keys.
- `ValueError`: `num_oov_buckets` is a negative integer.
- `ValueError`: `num_oov_buckets` and `default_value` are both specified.
- `ValueError`: if `dtype` is not integer or string.

Last updated November 2, 2017.