'''

Contains implementations of functions to read input data.

'''

from .dataset import stream\_block\_format\_dataset

import tensorflow.compat.v1 as tf

def data\_record\_input\_fn(

files, batch\_size, parse\_fn,

num\_threads=2, repeat=False, dataset\_fn=None,

keep\_rate=None, parts\_downsampling\_rate=None,

shards=None, shard\_index=None, shuffle=True, shuffle\_files=True, interleave=True,

initializable=False, log\_tf\_data\_summaries=False,

\*\*kwargs):

"""

Returns a nested structure of tf.Tensors containing the next element.

Used by ``train\_input\_fn`` and ``eval\_input\_fn`` in DataRecordTrainer.

By default, works with DataRecord dataset for compressed partition files.

Args:

files:

List of files that will be parsed.

batch\_size:

number of samples per batch.

parse\_fn:

function passed to data loading for parsing individual data records.

Usually one of the decoder functions like ``parsers.get\_sparse\_parse\_fn``.

num\_threads (optional):

number of threads used for loading data. Defaults to 2.

repeat (optional):

Repeat the dataset indefinitely. Defaults to False.

Useful when you want to use ``train\_steps`` or ``eval\_steps``

greater than the size of the dataset

(otherwise Estimator.[train,evaluate] stops when the end of the dataset is reached).

dataset\_fn (optional):

A function that modifies the dataset after it reads different interleaved parts files.

Defaults to:

.. code-block:: python

def dataset\_fn(dataset, parse\_fn, batch\_size):

return dataset.batch(batch\_size).map(parse\_fn, 1)

keep\_rate (optional):

A float value in (0.0, 1.0] that indicates to drop records according to the Bernoulli

distribution with p = 1 - keep\_rate.

Defaults to None (no records dropped).

parts\_downsampling\_rate (optional):

A float value in (0.0, 1.0] that indicates the factor by which to downsample part files.

For example, a value of 0.2 means only 20 percent of part files become part of the dataset.

shards (optional):

Number of partitions to shard the dataset into. This is useful for codistillation

(https://arxiv.org/pdf/1804.03235.pdf) and other techniques that require each worker to

train on disjoint partitions of the dataset.

The dataset is not sharded by default.

shard\_index (optional):

Which partition of the dataset to use if ``shards`` is set.

shuffle (optional):

Whether to shuffle the records. Defaults to True.

shuffle\_files (optional):

Shuffle the list of files. Defaults to True.

When False, files are iterated in the order they are passed in.

interleave (optional):

Interleave records from multiple files in parallel. Defaults to True.

initializable (optional):

A boolean indicator. When the Dataset Iterator depends on some resource, e.g. a HashTable or

a Tensor, i.e. it's an initializable iterator, set it to True. Otherwise, default value (false)

is used for most plain iterators.

log\_tf\_data\_summaries (optional):

A boolean indicator denoting whether to add a `tf.data.experimental.StatsAggregator` to the

tf.data pipeline. This adds summaries of pipeline utilization and buffer sizes to the output

events files. This requires that `initializable` is `True` above.

Returns:

Iterator of elements of the dataset.

"""

if not parse\_fn:

raise ValueError("default\_input\_fn requires a parse\_fn")

if log\_tf\_data\_summaries and not initializable:

raise ValueError("Require `initializable` if `log\_tf\_data\_summaries`.")

dataset = stream\_block\_format\_dataset(

files=files,

parse\_fn=parse\_fn,

batch\_size=batch\_size,

repeat=repeat,

num\_threads=num\_threads,

dataset\_fn=dataset\_fn,

keep\_rate=keep\_rate,

parts\_downsampling\_rate=parts\_downsampling\_rate,

shards=shards,

shard\_index=shard\_index,

shuffle=shuffle,

shuffle\_files=shuffle\_files,

interleave=interleave,

\*\*kwargs

)

# Add a tf.data.experimental.StatsAggregator

# https://www.tensorflow.org/versions/r1.15/api\_docs/python/tf/data/experimental/StatsAggregator

if log\_tf\_data\_summaries:

aggregator = tf.data.experimental.StatsAggregator()

options = tf.data.Options()

options.experimental\_stats.aggregator = aggregator

dataset = dataset.with\_options(options)

stats\_summary = aggregator.get\_summary()

tf.add\_to\_collection(tf.GraphKeys.SUMMARIES, stats\_summary)

if initializable:

# when the data parsing dpends on some HashTable or Tensor, the iterator is initalizable and

# therefore we need to be run explicitly

iterator = dataset.make\_initializable\_iterator()

tf.add\_to\_collection(tf.GraphKeys.TABLE\_INITIALIZERS, iterator.initializer)

else:

iterator = dataset.make\_one\_shot\_iterator()

return iterator.get\_next()

default\_input\_fn = data\_record\_input\_fn # pylint: disable=invalid-name