Package 'tfio'

December 19, 2019

Type Package

Title Interface to 'TensorFlow IO'

Version 0.4.1

Description
Interface to 'TensorFlow IO', Datasets and filesystem extensions maintained by `TensorFlow SIG-IO` <https://github.com/tensorflow/community/blob/master/sigs/io/CHARTER.md>.

License Apache License 2.0

URL https://github.com/tensorflow/io

BugReports https://github.com/tensorflow/io/issues

SystemRequirements TensorFlow >= 1.13.0 (https://www.tensorflow.org/) and TensorFlow IO >= 0.4.0 (https://github.com/tensorflow/io)

Encoding UTF-8

LazyData true

Depends R (>= 3.1)

Imports reticulate (>= 1.10), tensorflow (>= 1.9), tfdatasets (>= 1.9), forge, magrittr

RoxygenNote 7.0.2

Suggests testthat, knitr

VignetteBuilder knitr

NeedsCompilation no

Author TensorFlow IO Contributors [aut, cph] (Full list of contributors can be found at <https://github.com/tensorflow/io/graphs/contributors>),
Yuan Tang [aut, cre] (<https://orcid.org/0000-0001-5243-233X>),
TensorFlow Authors [cph],
Ant Financial [cph],
RStudio [cph]

Maintainer Yuan Tang <terrytangyuan@gmail.com>

Repository CRAN

Date/Publication 2019-12-19 16:50:02 UTC

R topics documented:

`arrow_feather_dataset` *Creates a* `ArrowFeatherDataset`.

Description

An Arrow Dataset for reading record batches from Arrow feather files. Feather is a light-weight columnar format ideal for simple writing of Pandas DataFrames.

Usage

```
arrow_feather_dataset(filenames, columns, output_types, output_shapes = NULL)
```

Arguments

`filenames`	A `tf.string` tensor, list or scalar containing files in Arrow Feather format.
`columns`	A list of column indices to be used in the Dataset.
`output_types`	Tensor dtypes of the output tensors.
`output_shapes`	TensorShapes of the output tensors or `NULL` to infer partial.

Examples

```
## Not run:
dataset <- arrow_feather_dataset(
    list('/path/to/a.feather', '/path/to/b.feather'),
    columns = reticulate::tuple(0L, 1L),
    output_types = reticulate::tuple(tf$int32, tf$float32),
    output_shapes = reticulate::tuple(list(), list())) %>%
  dataset_repeat(1)

sess <- tf$Session()
iterator <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iterator)

until_out_of_range({
  batch <- sess$run(next_batch)
  print(batch)
})

## End(Not run)
```

---

arrow_stream_dataset *Creates a* ArrowStreamDataset.

Description

An Arrow Dataset for reading record batches from an input stream. Currently supported input streams are a socket client or stdin.

Usage

```
arrow_stream_dataset(host, columns, output_types, output_shapes = NULL)
```

Arguments

host	A `tf.string` tensor or string defining the input stream. For a socket client, use "<HOST_IP>:<PORT>", for stdin use "STDIN".
columns	A list of column indices to be used in the Dataset.
output_types	Tensor dtypes of the output tensors.
output_shapes	TensorShapes of the output tensors or `NULL` to infer partial.

Examples

```
## Not run:
dataset <- arrow_stream_dataset(
    host,
    columns = reticulate::tuple(0L, 1L),
    output_types = reticulate::tuple(tf$int32, tf$float32),
```

```
    output_shapes = reticulate::tuple(list(), list())) %>%
  dataset_repeat(1)

sess <- tf$Session()
iterator <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iterator)

until_out_of_range({
  batch <- sess$run(next_batch)
  print(batch)
})

## End(Not run)
```

`from_schema` *Create an Arrow Dataset from the given Arrow schema.*

Description

Infer output types and shapes from the given Arrow schema and create an Arrow Dataset.

Usage

```
from_schema(object, ...)
```

Arguments

`object`	An R object.
`...`	Optional arguments passed on to implementing methods.

`from_schema.arrow_feather_dataset` *Create an Arrow Dataset for reading record batches from Arrow feather files, inferring output types and shapes from the given Arrow schema.*

Description

Create an Arrow Dataset for reading record batches from Arrow feather files, inferring output types and shapes from the given Arrow schema.

Usage

```
## S3 method for class 'arrow_feather_dataset'
from_schema(object, schema, columns = NULL, host = NULL, filenames = NULL, ...)
```

Arguments

object	An R object.
schema	Arrow schema defining the record batch data in the stream.
columns	A list of column indices to be used in the Dataset.
host	Not used.
filenames	A tf.string tensor, list or scalar containing files in Arrow Feather format.
...	Optional arguments passed on to implementing methods.

---

from_schema.arrow_stream_dataset

Create an Arrow Dataset from an input stream, inferring output types and shapes from the given Arrow schema.

Description

Create an Arrow Dataset from an input stream, inferring output types and shapes from the given Arrow schema.

Usage

```
## S3 method for class 'arrow_stream_dataset'
from_schema(object, schema, columns = NULL, host = NULL, filenames = NULL, ...)
```

Arguments

object	An R object.
schema	Arrow schema defining the record batch data in the stream.
columns	A list of column indices to be used in the Dataset.
host	A tf.string tensor or string defining the input stream. For a socket client, use "<HOST_IP>:<PORT>", for stdin use "STDIN".
filenames	Not used.
...	Optional arguments passed on to implementing methods.

`ignite_dataset` *Create a* `IgniteDataset`.

Description

Apache Ignite is a memory-centric distributed database, caching, and processing platform for transactional, analytical, and streaming workloads, delivering in-memory speeds at petabyte scale. This contrib package contains an integration between Apache Ignite and TensorFlow. The integration is based on tf.data from TensorFlow side and Binary Client Protocol from Apache Ignite side. It allows to use Apache Ignite as a datasource for neural network training, inference and all other computations supported by TensorFlow. Ignite Dataset is based on Apache Ignite Binary Client Protocol.

Usage

```
ignite_dataset(
  cache_name,
  host = "localhost",
  port = 10800,
  local = FALSE,
  part = -1,
  page_size = 100,
  username = NULL,
  password = NULL,
  certfile = NULL,
  keyfile = NULL,
  cert_password = NULL
)
```

Arguments

`cache_name`	Cache name to be used as datasource.
`host`	Apache Ignite Thin Client host to be connected.
`port`	Apache Ignite Thin Client port to be connected.
`local`	Local flag that defines to query only local data.
`part`	Number of partitions to be queried.
`page_size`	Apache Ignite Thin Client page size.
`username`	Apache Ignite Thin Client authentication username.
`password`	Apache Ignite Thin Client authentication password.
`certfile`	File in PEM format containing the certificate as well as any number of CA certificates needed to establish the certificate's authenticity.
`keyfile`	File containing the private key (otherwise the private key will be taken from certfile as well).
`cert_password`	Password to be used if the private key is encrypted and a password is necessary.

Examples

```
## Not run:
dataset <- ignite_dataset(
    cache_name = "SQL_PUBLIC_TEST_CACHE", port = 10800) %>%
  dataset_repeat(1)

sess <- tf$Session()
iterator <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iterator)

until_out_of_range({
  batch <- sess$run(next_batch)
  print(batch)
})

## End(Not run)
```

---

kafka_dataset *Creates a* KafkaDataset.

Description

Creates a KafkaDataset.

Usage

```
kafka_dataset(
  topics,
  servers = "localhost",
  group = "",
  eof = FALSE,
  timeout = 1000
)
```

Arguments

topics	A tf.string tensor containing one or more subscriptions, in the format of [topic:partition:offset:length], by default length is -1 for unlimited.
servers	A list of bootstrap servers.
group	The consumer group id.
eof	If True, the kafka reader will stop on EOF.
timeout	The timeout value for the Kafka Consumer to wait (in millisecond).

Examples

```
## Not run:
dataset <- kafka_dataset(
    topics = list("test:0:0:4"), group = "test", eof = TRUE) %>%
  dataset_repeat(1)

sess <- tf$Session()
iterator <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iterator)

until_out_of_range({
  batch <- sess$run(next_batch)
  print(batch)
})

## End(Not run)
```

---

kinesis_dataset *Creates a* KinesisDataset.

Description

Kinesis is a managed service provided by AWS for data streaming. This dataset reads messages from Kinesis with each message presented as a `tf.string`.

Usage

```
kinesis_dataset(stream, shard = "", read_indefinitely = TRUE, interval = 1e+05)
```

Arguments

`stream`	A `tf.string` tensor containing the name of the stream.
`shard`	A `tf.string` tensor containing the id of the shard.
`read_indefinitely`	If `True`, the Kinesis dataset will keep retry again on `EOF` after the `interval` period. If `False`, then the dataset will stop on `EOF`. The default value is `True`.
`interval`	The interval for the Kinesis Client to wait before it tries to get records again (in millisecond).

lmdb_dataset *Create a* LMDBDataset.

Description

This function allows a user to read data from a LMDB file. A lmdb file consists of (key value) pairs sequentially.

Usage

```
lmdb_dataset(filenames)
```

Arguments

filenames	A tf.string tensor containing one or more filenames.

Examples

```
## Not run:
dataset <- sequence_file_dataset("testdata/data.mdb") %>%
  dataset_repeat(1)

sess <- tf$Session()
iterator <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iterator)

until_out_of_range({
  batch <- sess$run(next_batch)
  print(batch)
})

## End(Not run)
```

make_libsvm_dataset *Create a Dataset from LibSVM files.*

Description

Create a Dataset from LibSVM files.

Usage

```
make_libsvm_dataset(
  file_names,
  num_features,
  dtype = NULL,
  label_dtype = NULL,
  batch_size = 1,
  compression_type = "",
  buffer_size = NULL,
  num_parallel_parser_calls = NULL,
  drop_final_batch = FALSE,
  prefetch_buffer_size = 0
)
```

Arguments

file_names	A tf.string tensor containing one or more filenames.
num_features	The number of features.
dtype	The type of the output feature tensor. Default to tf.float32.
label_dtype	The type of the output label tensor. Default to tf.int64.
batch_size	An integer representing the number of records to combine in a single batch, default 1.
compression_type	A tf.string scalar evaluating to one of "" (no compression), "ZLIB", or "GZIP".
buffer_size	A tf.int64 scalar denoting the number of bytes to buffer. A value of 0 results in the default buffering values chosen based on the compression type.
num_parallel_parser_calls	Number of parallel records to parse in parallel. Defaults to an automatic selection.
drop_final_batch	Whether the last batch should be dropped in case its size is smaller than batch_size; the default behavior is not to drop the smaller batch.
prefetch_buffer_size	An integer specifying the number of feature batches to prefetch for performance improvement. Defaults to auto-tune. Set to 0 to disable prefetching.

---

mnist_image_dataset *Creates a* MNISTImageDataset.

Description

This creates a dataset for MNIST images.

Usage

```
mnist_image_dataset(filenames, compression_type = NULL)
```

Arguments

filenames	A tf.string tensor containing one or more filenames.
compression_type	A tf.string scalar evaluating to one of "" (no compression), "ZLIB", or "GZIP".

---

mnist_label_dataset *Creates a* MNISTLabelDataset.

Description

This creates a dataset for MNIST labels.

Usage

```
mnist_label_dataset(filenames, compression_type = NULL)
```

Arguments

filenames	A tf.string tensor containing one or more filenames.
compression_type	A tf.string scalar evaluating to one of "" (no compression), "ZLIB", or "GZIP".

---

parquet_dataset *Create a* ParquetDataset.

Description

This allows a user to read data from a parquet file.

Usage

```
parquet_dataset(filenames, columns, output_types)
```

Arguments

filenames	A 0-D or 1-D tf.string tensor containing one or more filenames.
columns	A 0-D or 1-D tf.int32 tensor containing the columns to extract.
output_types	A tuple of tf.DType objects representing the types of the columns returned.

Examples

```
## Not run:
dtypes <- tf$python$framework$dtypes
output_types <- reticulate::tuple(
  dtypes$bool, dtypes$int32, dtypes$int64, dtypes$float32, dtypes$float64)
dataset <- parquet_dataset(
    filenames = list("testdata/parquet_cpp_example.parquet"),
    columns = list(0, 1, 2, 4, 5),
    output_types = output_types) %>%
  dataset_repeat(2)

sess <- tf$Session()
iterator <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iterator)

until_out_of_range({
  batch <- sess$run(next_batch)
  print(batch)
})

## End(Not run)
```

---

pubsub_dataset *Creates a* PubSubDataset.

Description

This creates a dataset for consuming PubSub messages.

Usage

```
pubsub_dataset(subscriptions, server = NULL, eof = FALSE, timeout = 1000)
```

Arguments

subscriptions	A `tf.string` tensor containing one or more subscriptions.
server	The pubsub server.
eof	If True, the pubsub reader will stop on EOF.
timeout	The timeout value for the PubSub to wait (in millisecond).

sequence_file_dataset *Create a* SequenceFileDataset.

Description

This function allows a user to read data from a hadoop sequence file. A sequence file consists of (key value) pairs sequentially. At the moment, `org.apache.hadoop.io.Text` is the only serialization type being supported, and there is no compression support.

Usage

```
sequence_file_dataset(filenames)
```

Arguments

`filenames`	A `tf.string` tensor containing one or more filenames.

Examples

```
## Not run:
dataset <- sequence_file_dataset("testdata/string.seq") %>%
  dataset_repeat(1)

sess <- tf$Session()
iterator <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iterator)

until_out_of_range({
  batch <- sess$run(next_batch)
  print(batch)
})

## End(Not run)
```

tfio *TensorFlow IO API for R*

Description

This library provides an R interface to the TensorFlow IO API that provides datasets and filesystem extensions maintained by SIG-IO.

tiff_dataset *Create a* TIFFDataset.

Description

A TIFF Image File Dataset that reads the TIFF file.

Usage

```
tiff_dataset(filenames)
```

Arguments

filenames	A tf.string tensor containing one or more filenames.

Examples

```
## Not run:
dataset <- tiff_dataset(
    filenames = list("testdata/small.tiff")) %>%
  dataset_repeat(1)

sess <- tf$Session()
iterator <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iterator)

until_out_of_range({
  batch <- sess$run(next_batch)
  print(batch)
})

## End(Not run)
```

video_dataset *Create a* VideoDataset *that reads the video file.*

Description

This allows a user to read data from a video file with ffmpeg. The output of VideoDataset is a sequence of (height, weight, 3) tensor in rgb24 format.

Usage

```
video_dataset(filenames)
```

Arguments

filenames	A tf.string tensor containing one or more filenames.

Examples

```
## Not run:
dataset <- video_dataset(
    filenames = list("testdata/small.mp4")) %>%
  dataset_repeat(2)

sess <- tf$Session()
iterator <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iterator)

until_out_of_range({
  batch <- sess$run(next_batch)
  print(batch)
})

## End(Not run)
```

---

webp_dataset *Create a* WebPDataset.

Description

A WebP Image File Dataset that reads the WebP file.

Usage

```
webp_dataset(filenames)
```

Arguments

filenames	A tf.string tensor containing one or more filenames.

Examples

```
## Not run:
dataset <- webp_dataset(
    filenames = list("testdata/sample.webp")) %>%
  dataset_repeat(1)

sess <- tf$Session()
iterator <- make_iterator_one_shot(dataset)
next_batch <- iterator_get_next(iterator)

until_out_of_range({
```

```
  batch <- sess$run(next_batch)
  print(batch)
})

## End(Not run)
```

Index