Package 'sparktf'

July 23, 2025

Type Package

Title Interface for 'TensorFlow' 'TFRecord' Files with 'Apache Spark'

Version 0.1.0

Description A 'sparklyr' extension that enables reading and writing 'TensorFlow' TFRecord files via 'Apache Spark'.

License Apache License (>= 2.0)

Encoding UTF-8

SystemRequirements TensorFlow (https://www.tensorflow.org/)

LazyData true

Depends R (>= 3.1.2)

Imports sparklyr (>= 1.0)

RoxygenNote 6.1.0

Suggests testthat, dplyr

NeedsCompilation no

Author Kevin Kuo [aut, cre] (ORCID: <https://orcid.org/0000-0001-7803-7901>)

Maintainer Kevin Kuo <kevin.kuo@rstudio.com>

Repository CRAN

Date/Publication 2019-03-05 14:30:03 UTC

Contents

spark_read_tfrecord *Read a TFRecord File*

Description

Read a TFRecord file as a Spark DataFrame.

Usage

```
spark_read_tfrecord(sc, name = NULL, path = name, schema = NULL,
  record_type = c("Example", "SequenceExample"), overwrite = TRUE)
```

Arguments

sc	A spark conneciton.
name	The name to assign to the newly generated table or the path to the file. Note that if a path is provided for the ‘name‘ argument then one cannot specify a name.
path	The path to the file. Needs to be accessible from the cluster. Supports the "hdfs://", "s3a://" and "file://" protocols.
schema	(Currently unsupported.) Schema of TensorFlow records. If not provided, the schema is inferred from TensorFlow records.
record_type	Input format of TensorFlow records. By default it is Example.
overwrite	Boolean; overwrite the table with the given name if it already exists?

Examples

```
## Not run:
iris_tbl <- copy_to(sc, iris)
data_path <- file.path(tempdir(), "iris")
df1 <- iris_tbl %>%
ft_string_indexer_model(
  "Species", "label",
  labels = c("setosa", "versicolor", "virginica")
)

df1 %>%
spark_write_tfrecord(
  path = data_path,
  write_locality = "local"
)

spark_read_tfrecord(sc, data_path)

## End(Not run)
```

spark_write_tfrecord *Write a Spark DataFrame to a TFRecord file*

Description

Serialize a Spark DataFrame to the TensorFlow TFRecord format for training or inference.

Usage

```
spark_write_tfrecord(x, path, record_type = c("Example",
  "SequenceExample"), write_locality = c("distributed", "local"),
  mode = NULL)
```

Arguments

x	A Spark DataFrame
path	The path to the file. Needs to be accessible from the cluster. Supports the "hdfs://", "s3a://", and "file://" protocols.
record_type	Output format of TensorFlow records. One of "Example" and "SequenceExample".
write_locality	Determines whether the TensorFlow records are written locally on the workers or on a distributed file system. One of "distributed" and "local". See Details for more information.
mode	A character element. Specifies the behavior when data or table already exists. Supported values include: 'error', 'append', 'overwrite' and 'ignore'. Notice that 'overwrite' will also change the column structure. For more details see also http://spark.apache.org/docs/latest/sql-programming-guide.html#save-modes for your version of Spark.

Details

For write_locality = local, each of the workers stores on the local disk a subset of the data. The subset that is stored on each worker is determined by the partitioning of the DataFrame. Each of the partitions is coalesced into a single TFRecord file and written on the node where the partition lives. This is useful in the context of distributed training, in which each of the workers gets a subset of the data to work on. When this mode is activated, the path provided to the writer is interpreted as a base path that is created on each of the worker nodes, and that will be populated with data from the DataFrame.

Examples

```
## Not run:
iris_tbl <- copy_to(sc, iris)
data_path <- file.path(tempdir(), "iris")
df1 <- iris_tbl %>%
ft_string_indexer_model(
  "Species", "label",
```

```
  labels = c("setosa", "versicolor", "virginica")
)

df1 %>%
spark_write_tfrecord(
  path = data_path,
  write_locality = "local"
)

## End(Not run)
```

Index