Introduction to Data Pipelines

CLEANING DATA WITH PYSPARK

Mike Metzger
Data Engineering Consultant

What is a data pipeline?

- A set of steps to process data from source(s) to final output
- Can consist of any number of steps or components
- Can span many systems
- We will focus on data pipelines within Spark

What does a data pipeline look like?

- Input(s)
  - CSV, JSON, web services, databases
- Transformations
  - `withColumn()` , `.filter()` , `.drop()`
- Output(s)
  - CSV, Parquet, database
- Validation
- Analysis

Pipeline details

- Not formally defined in Spark
- Typically all normal Spark code required for task

```
schema = StructType([
StructField('name', StringType(), False),
StructField('age', StringType(), False)
])
df = spark.read.format('csv').load('datafile').schema(schema)
df = df.withColumn('id', monotonically_increasing_id())
...
df.write.parquet('outdata.parquet')
df.write.json('outdata.json')
```

Let's Practice!

CLEANING DATA WITH PYSPARK

Data handling techniques

CLEANING DATA WITH PYSPARK

Mike Metzger
Data Engineering Consultant

What are we trying to parse?

- Incorrect data
  - Empty rows
  - Commented lines
  - Headers
- Nested structures
  - Multiple delimiters
- Non-regular data
  - Differing numbers of columns per row
- Focused on CSV data

```
width, height, image


# This is a comment
```

```
200    300    affenpinscher;0
```

```
600    450    Collie;307    Collie;101
600    449    Japanese_spaniel;23
```

Stanford ImageNet annotations

- Identifies dog breeds in images
- Provides list of all identified dogs in image
- Other metadata (base folder, image size, etc.)

Example rows:

```
02111277    n02111277_3206    500    375    Newfoundland,110,73,416,298
02108422    n02108422_4375    500    375    bull_mastiff,101,90,214,356 \
 bull_mastiff,282,74,416,370
```

Removing blank lines, headers, and comments

Spark's CSV parser:

- Automatically removes blank lines
- Can remove comments using an optional argument

```
df1 = spark.read.csv('datafile.csv.gz', comment='#')
```

- Handles header fields
  - Defined via argument
  - Ignored if a schema is defined

```
df1 = spark.read.csv('datafile.csv.gz', header='True')
```

Automatic column creation

Spark will:

- Automatically create columns in a DataFrame based on `sep` argument

```
df1 = spark.read.csv('datafile.csv.gz', sep=',')
```

- Defaults to using `,`
- Can still successfully parse if `sep` is not in string

```
df1 = spark.read.csv('datafile.csv.gz', sep='*')
```

- Stores data in column defaulting to `_c0`
- Allows you to properly handle nested separators

Let's practice!

CLEANING DATA WITH PYSPARK

Data validation

CLEANING DATA WITH PYSPARK

Mike Metzger
Data Engineering Consultant

Definition

Validation is:

- Verifying that a dataset complies with the expected format
- Number of rows / columns
- Data types
- Complex validation rules

Validating via joins

- Compares data against known values
- Easy to find data in a given set
- Comparatively fast

```
parsed_df = spark.read.parquet('parsed_data.parquet')
company_df = spark.read.parquet('companies.parquet')
verified_df = parsed_df.join(company_df, parsed_df.company == company_df.company)
```

This *automatically* removes any rows with a company not in the `valid_df` !

Complex rule validation

Using Spark components to validate logic:

- Calculations
- Verifying against external source
- Likely uses a UDF to modify / verify the DataFrame

Let's practice!

CLEANING DATA WITH PYSPARK

Final analysis and delivery

CLEANING DATA WITH PYSPARK

Mike Metzger
Data Engineering Consultant

Analysis calculations (UDF)

Calculations using UDF

```
def getAvgSale(saleslist):
  totalsales = 0
  count = 0
  for sale in saleslist:
    totalsales += sale[2] + sale[3]
    count += 2
  return totalsales / count

udfGetAvgSale = udf(getAvgSale, DoubleType())
df = df.withColumn('avg_sale', udfGetAvgSale(df.sales_list))
```

Analysis calculations (inline)

Inline calculations

```
df = df.read.csv('datafile')

df = df.withColumn('avg', (df.total_sales / df.sales_count))

df = df.withColumn('sq_ft', df.width * df.length)

df = df.withColumn('total_avg_size', udfComputeTotal(df.entries) / df.numEntries)
```

Let's practice!

CLEANING DATA WITH PYSPARK

Congratulations and next steps

CLEANING DATA WITH PYSPARK

Mike Metzger
Data Engineering Consultant

Next Steps

- Review Spark documentation
- Try working with data on actual clusters
- Work with various datasets

Thank you!

CLEANING DATA WITH PYSPARK