
Title: Filter Disease-Target Association with DOID

Purpose

This script processes a dataset of disease-target associations (`disease_gene_association.parquet`) by filtering records based on a specific DOID number. It outputs both a filtered dataset and an aggregated summary of metrics associated with the DOID. The script is designed to handle large datasets efficiently using Apache Spark.

Key Features

- Filters disease-target association data by a specific **DOID prefix**.
- Aggregates metrics for the filtered data.
- Saves results as separate Parquet files for downstream analysis.

Requirements

- **Apache Spark**: Ensure PySpark is installed and properly configured.
- **Input File**:
  - Path: `/home/jeremiah/Documents/a_tictac_data/2025_version/tictac_output/disease_gene_association.parquet`
  - File Format: Parquet
  - Required Columns:
    - `doid_uniprot`, `nDiseases`, `nDrug`, `nStud`, `nPub`, `nStudyNewness`, `nPublicationWeighted`, `meanRankScore`, `gene_symbol`
- **Command-Line Argument**:
  - `DOID_number`: The numeric part of the DOID to filter by.

Workflow

Step 1: Initialize Spark Session

A Spark session is initialized with memory optimizations to handle large datasets efficiently:

- **Driver Memory**: 8 GB
- **Executor Memory**: 16 GB

- **Off-Heap Memory**: Enabled with an additional 8 GB
- **Shuffle Partitions**: Set to 200 for balanced workload.

```
spark = SparkSession.builder \
    .appName("Process DOID Files") \
    .config("spark.driver.memory", "8g") \
    .config("spark.executor.memory", "16g") \
    .config("spark.memory.offHeap.enabled", "true") \
    .config("spark.memory.offHeap.size", "8g") \
    .config("spark.sql.shuffle.partitions", "200") \
    .getOrCreate()
```

Step 2: Load Input File

The dataset is loaded from the specified Parquet file. Only necessary columns are selected to optimize memory usage.

```
df = spark.read.parquet(file_path).select(
    "doid_uniprot", "nDiseases", "nDrug", "nStud", "nPub", "nStudyNewness",
    "nPublicationWeighted", "meanRankScore", "gene_symbol"
)
```

Step 3: Extract and Filter by DOID Prefix

The `doid_uniprot` column is split to extract the `DOID_prefix`. The dataset is then filtered to include only rows matching the specified DOID prefix.

```
df = df.withColumn("DOID_prefix", split(col("doid_uniprot"), "_").getItem(0))
filtered_df = df.filter(col("DOID_prefix") == doid_prefix)
```

Step 4: Repartition Data

The filtered dataset is repartitioned into 100 partitions to optimize memory management and parallelism.

```
filtered_df = filtered_df.repartition(100)
```

Step 5: Save Filtered Data

The filtered data is saved to a Parquet file named `DOID_<DOID_number>_disease_target_associations.parquet`.

```
filtered_output_path = f"DOID_{doid_number}_disease_target_associations.parquet"
filtered_df.write.mode("overwrite").parquet(filtered_output_path)
```

Step 6: Aggregate Metrics

The filtered data is aggregated to compute average metrics for the DOID. Aggregated columns include:

- `avg_nDiseases`, `avg_nDrug`, `avg_nStud`, `avg_nPub`, `avg_nStudyNewness`, `avg_nPublicationWeighted`, `avg_meanRankScore`

```
aggregated_df = filtered_df.groupBy("DOID_prefix").agg(
    avg("nDiseases").alias("avg_nDiseases"),
    avg("nDrug").alias("avg_nDrug"),
    avg("nStud").alias("avg_nStud"),
    avg("nPub").alias("avg_nPub"),
    avg("nStudyNewness").alias("avg_nStudyNewness"),
    avg("nPublicationWeighted").alias("avg_nPublicationWeighted"),
    avg("meanRankScore").alias("avg_meanRankScore")
)
```

Step 7: Save Aggregated Metrics

The aggregated metrics are saved to a Parquet file named `DOID_<DOID_number>_metrics_summary.parquet`.

```
aggregated_output_path = f"DOID_{doid_number}_metrics_summary.parquet"
aggregated_df.write.mode("overwrite").parquet(aggregated_output_path)
```

Step 8: Stop Spark Session

The Spark session is stopped to release resources.

```
spark.stop()
```

Execution

1. Ensure the input file (`disease_gene_association.parquet`) exists at the specified path.
2. Run the script with the desired DOID number as a command-line argument:

```
python process_doid_files.py <DOID_number>
```

3. Example: `python process_doid_files.py 1234`

Output

1. **Filtered Data**:
    - **File Name**: `DOID_<DOID_number>_disease_target_associations.parquet`

    - **Content**: Rows matching the specified DOID prefix.
2. **Aggregated Metrics**:
    - **File Name**: `DOID_<DOID_number>_metrics_summary.parquet`
    - **Content**: Summary of average metrics for the specified DOID.

Performance Optimizations

- **Selective Column Loading**: Reads only necessary columns to minimize memory usage.
- **Repartitioning**: Optimizes memory utilization during processing.
- **Off-Heap Memory**: Enables additional memory allocation to handle large datasets.

Future Improvements

- Add error handling for invalid or missing input files.
- Dynamically fetch the input file path from a configuration file.
- Support additional aggregation methods (e.g., sum, min, max) for metrics.

This documentation provides a detailed overview of the script's functionality, requirements, and usage, ensuring ease of understanding and application.