Rare Pattern Mining using Spark

The application utilizes PySpark to perform frequent pattern mining, specifically targeting the identification and comparison of rare and common itemsets. The application is designed to process transactional data, identify frequent itemsets, and generate association rules. It also implements custom logic to find interesting rules based on Jaccard similarity between itemsets.

Environment Setup

- **Docker Image**: Apache Spark Docker image named "wonderful_mahavira".
- **Mounting Code Directory**: The code directory on the local system is mounted to the Docker container at `/data/` using a bind mount.
- **Root Access**: The container is accessed in root mode for administrative tasks.
- **Running the Script**: The main Python script is executed using Spark's submit script: `/opt/spark/bin/spark-submit main.py`.
- **Data Preprocessing**: Before running the main script, data is preprocessed using `filepreprocessing.py` to convert features to integers suitable for pattern mining.

Code Overview

1. **Spark Session Initialization**:
   - A Spark session is created with a specified application name and memory configurations.
2. **Data Loading**:
   - Transactional data is loaded from a specified path and processed to transform space-delimited strings into arrays of integers.
3. **Frequent Pattern Mining Setup**:
   - The FP-Growth algorithm is configured with minimum support and confidence thresholds to identify frequent itemsets and generate association rules.
4. **Itemset Support Calculation**:
   - Support for each itemset is calculated by dividing the count of each itemset by the total number of transactions.
5. **Cluster Definition**:
   - Itemsets are categorized into 'rare' and 'common' clusters based on their support values.
6. **Association Rule Filtering**:
   - Association rules are filtered for each cluster to find rules that meet the cluster support criteria.
7. **Similarity Calculation**:
   - A user-defined function (UDF) for Jaccard similarity is applied to compare antecedents of association rules across clusters.
8. **Interesting Rule Identification**:
   - Rules with high similarity in antecedents and different consequents between the two clusters are identified as interesting.
9. **Output**:
   - The interesting rules are displayed and also saved to a CSV file for further analysis.

Sana Ghani
MS - Data & Knowledge Engineering
Otto-von-Guericke-University
Magdeburg