Ripe Pumpkins - A Movie Review Aggregation service

BUAN 5315 01 23SQ Big Data Analysis | June 3, 2023 | Sri Lakshmi Mallipudi

Use Case Name: "Building a Movie Recommendation Service with Apache Spark."
Use Case URL: https://www.codementor.io/@jadianes/building-a-recommender-with-apache-spark-python-example-app-part1-du1083qbw.

Introduction:
This use case involves building a movie recommender system called Pumpkinmeter using Apache Spark. The project uses the MovieLens dataset provided by GroupLens Research to train the recommender system. The goal is to measure collaborative recommendations for millions of fans on Ripe Pumpkins, a movie review-aggregation service. The board of directors wants to understand the potential of the Pumpkinmeter score and its impact on the new initiative.

Dataset used:
Name: MovieLens Latest Datasets
Source Address: https://grouplens.org/datasets/movielens/
Summary: GroupLens Research has collected and made available rating datasets from the MovieLens website, which are widely used in recommendation systems. For this use case, we will use the latest datasets, including the "small" dataset with 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users last updated 09/2018, and the "full" dataset with 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users which includes tag genome data with 14 million relevance scores across 1,100 tags last updated 09/2018. These datasets include information such as movie IDs, user IDs, ratings given by users, and timestamps.

Technical Details:
This use case applies the following technologies:

1. PySpark, the Python API for Apache Spark, to write Spark applications using Python by providing a Pythonic interface to interact with Spark's distributed computing capabilities.
2. Apache Spark's MLlib's Machine learning library and utilities for data preprocessing, feature engineering, and model training.
3. Collaborative Filtering algorithm to make personalized recommendations based on the preferences of similar users using the collective wisdom of a large group of users.
4. Alternating Least Squares (ALS) optimization algorithm to minimize the prediction error by iterating alternates between updating user and item factors.

Spark's ALS implementation provides collaborative recommendation capabilities. The code is executed in a Jupyter Notebook environment, allowing for interactive analysis and visualization of the results, and uses the PySpark API to interact with Spark. The above technologies are used together to preprocess the dataset, train the collaborative filtering model using ALS, and generate personalized movie recommendations based on user ratings and preferences.

Debugging Details:
During the project, several challenges were encountered and overcome, including_

1. Data preprocessing: The MovieLens dataset required parsing and preprocessing to transform it into the appropriate format for training the recommender system.
2. Handling missing values: The dataset might contain missing values or incomplete information, which must be handled properly to avoid errors during training.
3. Performance optimization: Techniques such as caching intermediate results are employed to optimize performance. Caching commonly used data structures or RDDs in memory reduces the need for repeated computations, improving overall performance.
4. Hyperparameter tuning: The project applied hyperparameter tuning techniques, such as cross-validation, to find the optimal values for parameters in the Collaborative Filtering algorithm and enhance the quality of recommendations generated by the model.

The project successfully built a recommender system using Apache Spark's Collaborative Filtering algorithm, demonstrating the ability to handle large-scale datasets and generate accurate recommendations. The key features of coding practices in this project include modular code design, code reusability, efficient data processing using Spark's DataFrame API, and leveraging the distributed computing capabilities of Spark for scalability.

Results:
The Spark-based recommender system was evaluated using two test scenarios for two users. The scenarios involved filtering out movies with less than 25 ratings and less than 100 ratings from the entire dataset. The results showed personalized recommendations based on the users' ratings and preferences. For each user and scenario, the top 15 recommended movies were generated.

Insights:
The insights gained from this use case have several business implications and impacts. Some of the insights and actionable items include:

1. Personalized movie recommendations: The Pumpkinmeter score can provide personalized movie recommendations to individual customers, enhancing their movie-watching experience and increasing the likelihood of staying with Ripe Pumpkins' services.

2. Customer retention and satisfaction: By understanding individual customers' preferences, Ripe Pumpkins can tailor their movie recommendations and improve customer satisfaction, reducing the likelihood of customers switching to competitors' services.
3. Business growth: A robust and accurate recommendation engine like Pumpkinmeter can attract more customers, increase engagement, and drive revenue growth for Ripe Pumpkins.

References:

1. "Collaborative Filtering." Apache Spark Documentation. <https://spark.apache.org/docs/latest/ml-collaborative-filtering.html>
2. Jose A Dianes. "Building a Recommender with Apache Spark." Codementor.io. Sep 14. 2015. Jun 03. 2023. < https://www.codementor.io/@jadianes/building-a-recommender-with-apache-spark-python-example-app-part1-du1083qbw>
3. Pulkit Sharma. "Comprehensive Guide to build a Recommendation Engine from Scratch (in Python)." AnalyticsVidhya.com. Dec 23. 2020. Jun 03. 2023. <https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/>
4. Yuefeng Zhang. "Machine Learning for Building Recommender System in Python." TowardsDataScience.com. Aug 04. 2020. Jun 03. 2023. <https://towardsdatascience.com/machine-learning-for-building-recommender-system-in-python-9e4922dd7e97>

Appendix:

Fields of the Dataset:

- movieId: ID of the movie
- title: Title of the movie
- genres: Genres of the movie
- userId: ID of the user
- rating: Rating given by the user for the movie
- timestamp: Timestamp of the rating