Scalable Book Recommendation System Using Collaborative Filtering and Distributed Computing

Course: Mining Big Data
Student: Rohith Perumandla
ID: 2174629

Appendix:

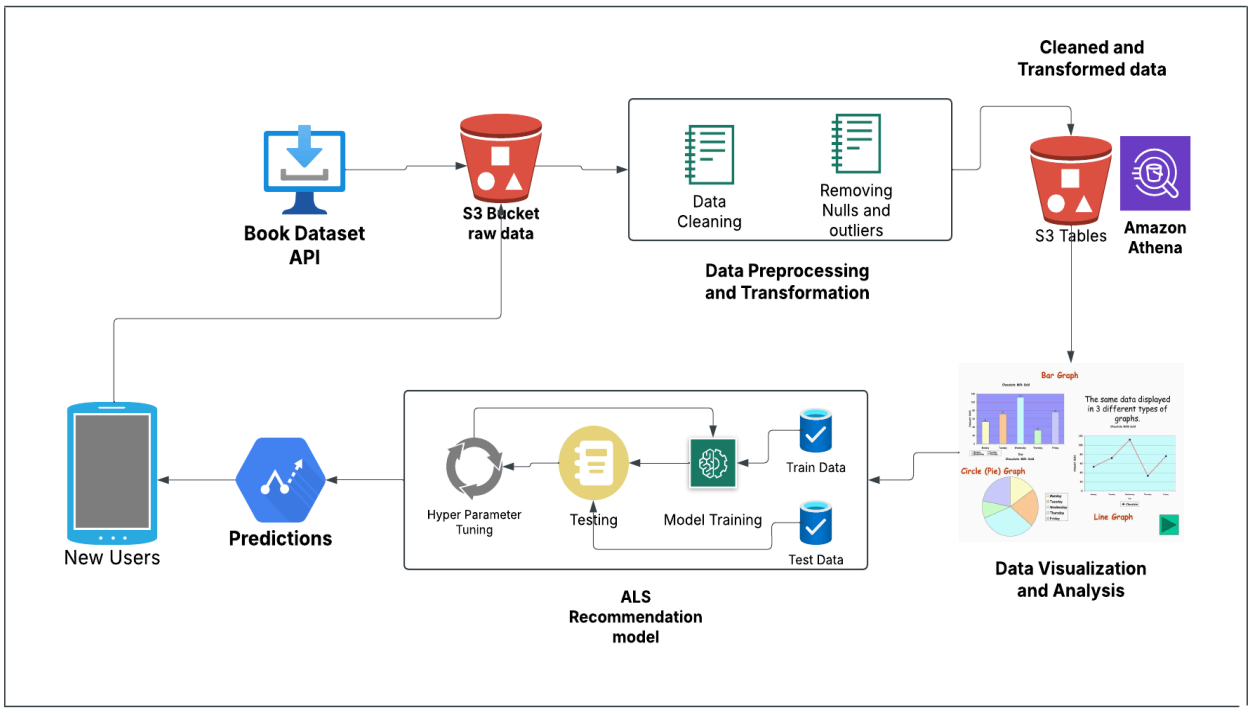
1. Abstract: Book recommendation systems are essential for enhancing user experience by suggesting relevant books based on individual preferences. This project develops a recommendation model using machine learning, specifically collaborative filtering techniques. Implemented with PySpark and utilizing AWS distributed computing for computation and Athena for querying the staging data, the system leverages user-item interaction data and matrix factorization to predict ratings. The model achieved a Root Mean Squared Error (RMSE) of 0.95, demonstrating its reliability in providing accurate book suggestions.

2. Data Source:

The dataset used in this project is sourced from the Book Genome Dataset, which is approximately 1.4 GB in size. This dataset includes various raw data such as book titles, authors, user ratings, reveiws and book-tag ratings. For this project, we specifically focused on two files: metadata.json and ratings.json. The metadata.json file contains detailed information about the books, including titles, authors, book IDs,

and external links, while the ratings.json file holds user ratings for the books. The dataset can be accessed at GroupLens Book Genome Dataset.

3. Machine Learning Pipeline Architecture:



PySpark

4. Machine Learning Pipeline.

1. **Installing Libraries and Importing.**
   All essential libraries required for the project have been successfully installed and imported. These libraries include key dependencies for distributed computing, data processing, and machine learning model development.

2. **Raw Data Storage in S3 Bucket:**
   The raw dataset is initially stored in an **Amazon S3 bucket**, providing a scalable and cost-efficient storage solution. This setup ensures the seamless integration of data into the first stage of the machine learning pipeline, which begins with data preprocessing.

3. **Data Preprocessing:**
   Data preprocessing is conducted using **PySpark DataFrames**, leveraging the power of **distributed computing** to enhance computational performance and reduce processing time. The dataset undergoes several stages of cleaning, including the removal of **null values**, **duplicates**,

and **outliers** (specifically limiting the ratings to the 0-5 range). These preprocessing steps are essential for ensuring high-quality data, free from inconsistencies or ambiguities, and make the dataset ready for accurate model training and evaluation.

4. **Store preprocessed data for Athena Querying:**
   Once the data is thoroughly cleaned, transformed, and preprocessed, it is stored in a database table optimized for efficient querying through Amazon Athena. Athena provides serverless, fast SQL queries on large datasets, enabling efficient exploration and analysis of the preprocessed data for further model development.

5. **Exploratory Data Analysis (EDA):**
   EDA is performed in two distinct ways to gain insights from the dataset:

   Using Athena for Querying and Visualization: Athena is employed to run SQL queries directly on the preprocessed dataset stored in the database. This allows for efficient querying of large datasets without the need to load the entire data into memory. Through SQL, we can extract useful information such as the top-rated and least-rated books, the most-read books, and other important statistical metrics. The results from these queries are then visualized by converting the query outputs into DataFrames for further analysis and presentation. This method leverages the flexibility of SQL for data exploration, while Athena optimizes query performance on large-scale datasets.

   Using PySpark DataFrame Techniques: Alternatively, the entire dataset can be imported into a PySpark DataFrame, where we apply distributed computing techniques to process and analyze the data. By leveraging PySpark's powerful DataFrame operations, we can perform more complex transformations, aggregations, and visualizations. This approach is beneficial when more advanced analysis or custom processing is required. For instance, we can compute the average ratings, identify the most popular books, and explore user-item interaction patterns to gain deeper insights into book preferences and user behavior. I use this method for analysing the data.

6. **ML Recommendation Training:**

   The dataset queried using Athena is used to train the recommendation model. For this task, I employ ALS (Alternating Least Squares), a matrix factorization technique that is well-suited for collaborative filtering problems like book recommendation systems. ALS is particularly effective in scenarios where we want to predict missing values in a user-item matrix, such as predicting book ratings based on user interactions. It works by decomposing the user-item interaction matrix into latent factors that represent user preferences and item characteristics, which are then used to predict ratings. I chose ALS because of its scalability and efficiency, especially in large datasets, and its ability to handle sparse matrices, which is common in recommendation systems.

   The model is built to predict book ratings on a continuous scale, ranging from 1 to 5, treating the problem as a regression task. To ensure the model generalizes well and avoids overfitting, I implemented a robust data-splitting strategy. The data is divided into training and validation sets. To further enhance the model's ability to generalize, cross-validation is employed during the training phase. Cross-validation helps assess the model's performance on different subsets of the training data, providing insights into its stability and reducing the risk of overfitting to any single portion of the dataset.

7. **Model Evaluation:**

Once the model is trained using cross-validation, it is then tested on the separate validation set to evaluate its performance. This testing phase ensures that the model has not learned spurious patterns and is capable of making accurate predictions on unseen data.

Since the problem involves predicting ratings on a continuous scale, the performance of the model is evaluated using Root Mean Square Error (RMSE rather than using Precision, Recall and F1 Score on K recommendations), which is an appropriate metric for regression tasks. RMSE measures the average magnitude of the error between predicted ratings and actual ratings. In this case, the model achieved an RMSE of approximately 0.95, indicating that the predicted ratings are within 1 point of the actual ratings on average. This demonstrates that the model performs well, making reliable book recommendations with good performance.

5. Conclusion and Future Work:

In conclusion, collaborative filtering has proven to be an effective and efficient algorithm for recommending books based on user history. By utilizing past user interactions and preferences, this approach allows us to predict books that users are likely to enjoy, enhancing their reading experience and making personalized recommendations. This helps users discover new books that align with their tastes, ultimately increasing engagement and satisfaction. The power of collaborative filtering lies in its ability to learn from user behaviors, offering dynamic and personalized suggestions that evolve as user preferences change.

One of the key advantages of this project is the use of **PySpark's distributed computing** capabilities. By leveraging PySpark, we were able to scale our computations effectively, processing more than **5 million rows of ratings data** quickly and efficiently. The distributed nature of PySpark enabled parallel processing, significantly improving the speed of data preprocessing, transformation, and model training. This was crucial given the large size of the dataset, as it would have been computationally expensive to handle such data on a single machine.

These technologies PySpark for distributed computing using m5.xlarge cluster, S3 for storage, and Athena for querying—worked together to make the project feasible and efficient, processing the large-scale dataset at speed. The ability to handle **over 5 million ratings data points** made this system scalable and capable of processing vast amounts of data with ease, allowing us to focus on refining the machine learning model itself.

Looking ahead, there are several exciting avenues for future work. Given that the data includes reviews from Goodreads, integrating sentiment analysis could further enhance the recommendation system by identifying the emotional tone behind user feedback and factoring it into the book suggestions. Additionally, incorporating cosine similarity techniques could allow for more sophisticated recommendations based on book genres, tags, titles, and descriptions. This would create a more holistic approach to recommendations, factoring in both user preferences and the inherent properties of the books themselves. Such a comprehensive system could lead to a much more personalized and context-aware book recommendation engine, and this idea holds the potential to be developed into a large-scale, complex project that could serve as a powerful tool for the publishing and book discovery industries.

6. Future System Architecture

The future system could leverage AWS Step Functions for automating the orchestration of the entire machine learning pipeline. This would include automated workflows for data ingestion, transformation, and cleaning, as well as periodic retraining of the model with the most up-to-date dataset. By utilizing AWS Step Functions, we can streamline the process of triggering data processing and model training tasks in a scalable and fault-tolerant manner. This architecture supports batch processing, enabling the system to efficiently handle large datasets and retrain the model periodically without manual intervention. The integration of automation ensures that the model remains up-to-date with fresh data, leading to improved recommendations and better performance over time.