Recommendation System Evaluation Report

Introduction:

In this report, we present an approach to evaluate the performance of a hybrid recommendation system that combines collaborative filtering, content-based filtering, and community detection techniques. The approach aims to provide users with more personalized and accurate product recommendations based on their interactions and preferences. We will describe the algorithms used, the data sources, the parameter choices, and discuss the strengths and limitations of the solution in detail.

Algorithms and Techniques Used:

Collaborative Filtering:

Collaborative filtering leverages the historical behavior and preferences of users to make recommendations. In this approach, a user-item matrix is created, where rows represent users and columns represent products. The Nearest Neighbors algorithm with cosine similarity is employed to find similar users, and their preferences are used to recommend products to the target user.

Content-Based Filtering:

Content-based filtering suggests items to users based on their past interactions with similar items. In our approach, we identify the user's top interest category based on weighted interest scores and recommend products within that category.

Community Detection:

The Louvain community detection algorithm is used to group users and products into communities based on their interactions. Users belonging to the same community are likely to have similar preferences.

Machine Learning Model:

We employ a Linear Regression model to combine recommendations from different sources. Features are created based on whether a product is recommended by collaborative filtering, content-based filtering, or community detection. The model learns to weigh these sources and predict the likelihood of a product being relevant.

Steps:

Loading Data: The code begins by loading data from several CSV files, including 'interest_scores.csv', 'product.csv', 'purchases.csv', 'interactions.csv', and 'user.csv', into respective DataFrames.

Data Preprocessing:

Duplicates Removal: Duplicate entries in the purchase data are removed based on 'User ID' and 'Product ID'.

Data Type Conversion and Filling Missing Values: Columns in 'interest_scores_df' are converted to numeric data types, and missing values are filled with 0.

Normalization:

Normalizing Interest Scores: The interest scores are normalized using the StandardScaler.

Collaborative Filtering:

User-Item Matrix: A user-item matrix is created from the purchase data.

Nearest Neighbors: NearestNeighbors algorithm with cosine similarity is used for collaborative filtering, based on the user-item matrix.

Content-Based Filtering:

Generating Recommendations: For a given user (specified as 'user_id'), a content-based recommendation is generated.

Top Interest Category: The user's top interest category is determined from the normalized interest scores.

Filtering Products: Products within the top interest category are filtered from the product DataFrame.

Content-Based Recommendations: Recommendations are generated based on the filtered products.

Louvain Clustering:

Graph Construction: A graph is constructed from interactions between users and products.

Louvain Community Detection: The Louvain community detection algorithm is used to group users and products into communities.

User Community: The community to which the specified user belongs is determined.

Community Products: Products within the same community as the user are identified.

Machine Learning Model for Recommendation Combination:

The code imports the LinearRegression model from the sklearn.linear_model module to create a machine learning model for recommendation combination.

It initializes two lists, X for feature matrices and y for target vectors, which will be used for training the model.

Creating Feature Vectors and Target Labels:

It retrieves the set of products that the user has purchased and treats them as relevant products.

Using the collaborative filtering (collab_recommendations), content-based (content_recommendations), and community-based (community_products) recommendations, the code constructs feature vectors for each product.

For each product, a feature vector is created with three values, indicating whether the product was recommended by collaborative filtering, content-based filtering, or community detection.

The target label (y-value) for each product is set to 1 if it's a relevant product (purchased by the user) and 0 otherwise.

Training the Model:

A LinearRegression model is trained using the feature matrices (X) and target labels (y).

Generating Combined Recommendations:

For each product recommended by any of the methods (collaborative filtering, content-based, community detection), a feature vector is created again with the same structure as before.

The trained model is then used to predict a score for each product based on its feature vector, which represents the likelihood of the user being interested in that product.

The products are stored as tuples containing the product ID and the predicted score.

Sorting and Selecting Top Recommendations:

The combined recommendations are sorted based on the predicted score in descending order.

The top N products with the highest scores are selected as the final recommendations.

Displaying Recommended Products:

The code retrieves the details of the recommended products from the product_df DataFrame based on their IDs.

It prints out the top N recommended products along with their IDs and names.

Evaluation:

Loading Interaction Data:

The code loads interaction data from the 'interactions.csv' file using the pd.read_csv() function.

Simulating Ground Truth Relevant Products:

The code simulates the ground truth relevant products for the user.

It identifies the products that the user has purchased and the products that the user has viewed (interaction type: 'Viewed').

The set of relevant products is obtained by taking the union of purchased and viewed products.

Simulated Recommended Products:

The code uses the set of recommended products generated earlier (recommended_products) as simulated recommended products.

Calculating Metrics:

True Positives (TP): The number of recommended products that were relevant to the user is calculated by finding the intersection between relevant products and simulated recommendations.

False Positives (FP): The number of recommended products that were not actually relevant (present in simulated recommendations but not in the relevant products).

False Negatives (FN): The number of relevant products that were not recommended (present in relevant products but not in simulated recommendations).

Precision: Precision is calculated as the ratio of true positives to the sum of true positives and false positives.

Recall: Recall is calculated as the ratio of true positives to the sum of true positives and false negatives.

F1-Score: The F1-Score is calculated using the harmonic mean of precision and recall. It's only calculated if the sum of precision and recall is greater than 0.

Results:

The calculated precision, recall, and F1-Score values are stored in the respective variables (precision, recall, and f1_score).

Data Sources:

Purchase Data: This dataset contains user-product interactions where users have purchased products. It's used to determine the user's purchased products and average purchase ratings for content-based filtering.

Interaction Data: This dataset captures various interactions between users and products, including viewing. It's used to simulate ground truth relevant products for evaluation.

Product Data: This dataset provides information about products, including their categories. It's used to identify top interest categories and relevant products for content-based filtering.

Recommended Products: These are the products recommended by the hybrid system, combining collaborative filtering, content-based filtering, and community detection.

Parameter Choices:

Nearest Neighbors: For collaborative filtering, the number of nearest neighbors (n_neighbors) is set to 6.

Linear Regression: A Linear Regression model is used to combine recommendations. Features are created based on the presence of recommendations from different sources.

Top N Recommendations: The top N products with the highest scores from the combined recommendations are selected as the final recommendations.

Strengths:

Personalization: The hybrid approach accounts for user preferences and behaviors from different angles, leading to more personalized recommendations.

Diversity: Combining multiple techniques enhances diversity in recommendations, reducing the risk of filter bubbles.

Adaptability: The solution adapts to changes in user behavior and interactions, ensuring relevance over time.

Limitations:

Cold Start: The system may struggle with new users who lack sufficient interaction history.

Scalability: As the number of users and products grows, the computational complexity of some techniques (e.g., collaborative filtering) increases.

Data Quality: Recommendations heavily rely on the quality of interaction data and the accuracy of item attributes.

Conclusion:

The hybrid recommendation system offers a comprehensive approach to provide users with accurate and personalized product recommendations. By combining collaborative filtering, content-based filtering, and community detection, the system addresses the limitations of individual techniques and enhances recommendation quality. However, the solution requires careful consideration of data quality and scalability as it scales to larger datasets. It offers a promising direction for creating more effective and personalized recommendation systems in the field of data science and engineering.