Product Recommendation Engine Documentation

Overview

This is the documentation for the Product Recommendation Engine project. The recommendation engine provides personalized product recommendations to users based on their preferences and historical interactions with products. The engine uses collaborative filtering and content-based filtering techniques to make relevant and accurate product recommendations.

Data

The dataset used for this project is sourced from Amazon and contains information about various products, user interactions, and product attributes. The dataset is loaded from a CSV file and preprocessed before building the recommendation models.

Data Preprocessing

The data preprocessing steps include the following:

1. Converting price values to numerical format: The `discounted_price` and `actual_price` columns are converted from strings containing currency symbols (₹) and commas to numerical format.

2. Handling missing values: In the `about_product` column, which contains the product descriptions, missing values are filled with empty strings.

3. Handling ratings: The `rating` column may contain multiple ratings separated by '|'. To handle this, the most frequent rating is selected and converted to a float. All ratings are then converted to floats for consistency.

Exploratory Data Analysis (EDA)

EDA is performed to gain insights into the dataset and understand the distribution of ratings and the number of ratings per user and product.

Distribution of Ratings

The distribution of ratings is visualized using a countplot, showing the frequency of each rating value.

Number of Ratings per User

A histogram is used to show the number of ratings given by each user. This helps in understanding user engagement.

Number of Ratings per Product

A histogram is used to show the number of ratings received by each product. This helps in understanding product popularity.

Visualizing Price Data

The prices of products are grouped into price ranges and visualized using histograms. This provides an overview of the price distribution.

Collaborative Filtering

Collaborative filtering is used to recommend products based on user-item interactions. Two collaborative filtering algorithms are used in this engine:

1. SVD (Singular Value Decomposition): SVD is a matrix factorization-based algorithm that decomposes the user-item interaction matrix into three matrices and predicts missing values.

2. KNNBasic with Item-Based Collaborative Filtering: This algorithm measures item-item similarity and predicts missing ratings based on the ratings of similar items.

The recommendation engine trains the collaborative filtering models using the training data and evaluates their performance using the test data.

Content-Based Filtering

Content-based filtering is used to recommend products based on their attributes. The product descriptions in the `about_product` column are used to calculate the similarity between products using TF-IDF (Term Frequency-Inverse Document Frequency) vectorization and cosine similarity.

Hybrid Recommendation

The final product recommendations are generated using a hybrid approach that combines collaborative filtering and content-based filtering.

1. Collaborative filtering recommendations using KNNBasic (Item-Based) are obtained.

2. Content-based filtering recommendations are obtained.

3. The results from both methods are combined, and duplicates are removed to produce the final recommendations.

Usage

To use the recommendation engine, the `hybrid_recommendations(user_id, product_id)` function is called, providing a user ID and a product ID as inputs. The function returns a list of recommended products for the user.

Example usage:

```
user_id = 'AG3D6O4STAQKAY2UVGEUV46KN35Q,AHMY5CWJMMK5BJRBBSNLYT3ONILA,AHCTC6U
product_id = 'B082LSVT4B'
recommendations = hybrid_recommendations(user_id, product_id)
if recommendations is not None:
    print("Hybrid Recommender Recommendations:")
    print(recommendations)
```

Real-Time Updates

Regarding real-time updates, the current implementation uses preprocessed data to generate recommendations. To achieve real-time updates, the recommendation process should incorporate new user interactions and product updates.

For real-time updates, consider the following steps:

1. Continuously collect and update user interactions and preferences: As users interact with products, their preferences and interactions should be captured and updated in the user-item interaction matrix.

2. Update collaborative filtering models: As new user-product interactions occur, the collaborative filtering models should be updated or retrained to consider the latest data.

3. Update content-based similarity matrix: The content-based filtering component relies on the similarity matrix calculated from product descriptions. Whenever new products are added or their descriptions change, the similarity matrix should be recalculated.

4. Implement a real-time data pipeline: Set up a data pipeline to handle real-time data updates and model training.

By incorporating these updates, the recommendation engine can provide real-time and up-to-date product recommendations to users based on their latest preferences and interactions.

Conclusion

The Product Recommendation Engine successfully combines collaborative filtering and content-based filtering techniques to provide personalized and relevant product recommendations to users. With the possibility of real-time updates, the engine can continuously adapt to user preferences and changing product attributes, ensuring an enhanced user experience.

For deployment, consider hosting the recommendation engine on a web application to provide a user-friendly interface for users to access and receive personalized product recommendations.