Sri Lanka Institute of Information and Technology

Information Retrieval and Web Analytics - IT3041

Personalized Product Recommendation System for E-commerce

Final Report 2024

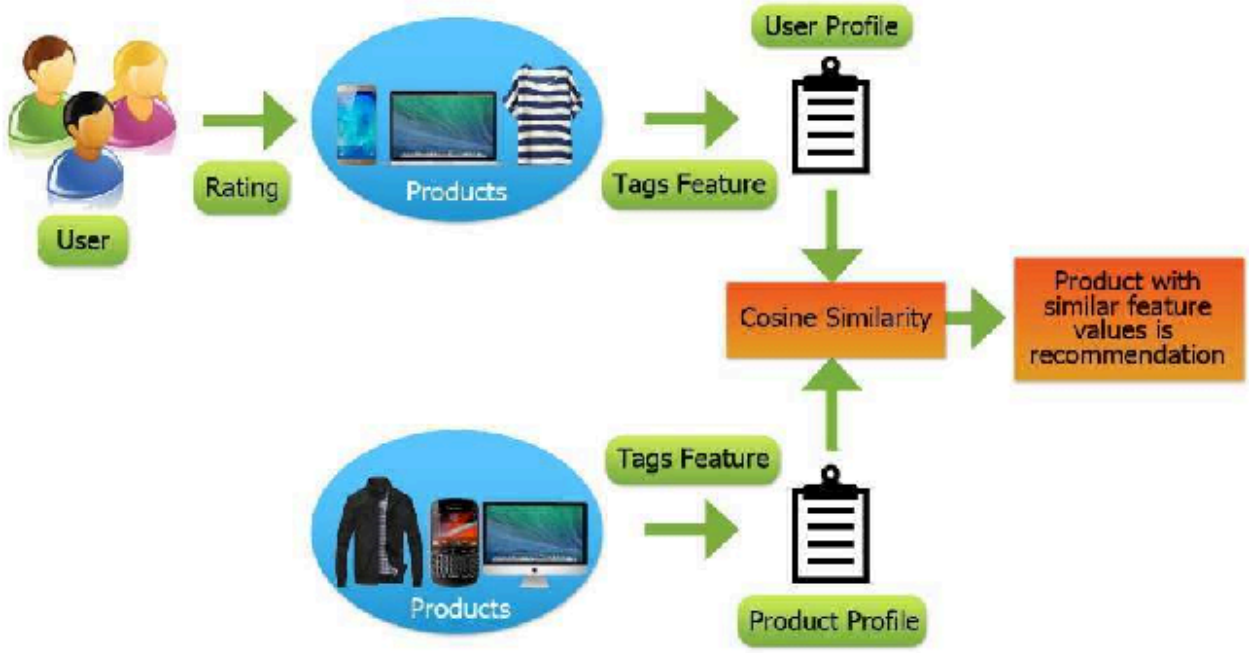
Name	IT Number
Gunasinghe S.N	IT22272690
Fernando B.T.D	IT22167446
Jayasooriya D.P.T.N	IT22257840

Table of content

Abstract

This project aims to develop a recommendation system for an e-commerce platform that suggests products based on user preferences, browsing history, and contextual factors such as time of day and seasonality. The objective is to enhance user experience by providing personalized product recommendations that improve customer engagement and increase sales.

The methodology includes implementing collaborative filtering to suggest products based on similar users' preferences, content-based filtering to recommend items with attributes similar to those users have previously interacted with, and a hybrid approach that combines both techniques for greater accuracy. The system utilizes user data, product features, and contextual information to generate relevant recommendations, with the architecture comprising a frontend, backend, and database to handle user interactions and product data efficiently.



Introduction

Background Information

With the rapid expansion of e-commerce platforms, the ability to provide personalized product recommendations has become essential for enhancing user experience and driving sales. E-commerce platforms deal with a vast array of products and a diverse user base, making it challenging to present relevant products to users. Recommendation systems, powered by data-driven algorithms, help to overcome this challenge by analyzing user preferences, browsing behavior, and contextual factors like seasonality and time of day.

Recommendation systems have evolved significantly, utilizing various techniques such as collaborative filtering, content-based filtering, and hybrid approaches to offer personalized suggestions. These systems can significantly impact a business's conversion rate by showcasing products that users are more likely to purchase, thereby improving customer satisfaction and increasing revenue.

Objective

The objective of this project is to develop a robust recommendation system for an e-commerce platform that suggests products based on:

1. User Preferences: Recommending products users have shown interest in through previous interactions, such as browsing or purchases.
2. Browsing History: Analyzing products that users have recently viewed to generate recommendations.
3. Contextual Factors: Incorporating real-time factors such as trending products to provide more relevant recommendations.

The system aims to improve customer engagement by offering personalized product suggestions, thus enhancing the overall shopping experience on the platform. By integrating multiple recommendation techniques, the project seeks to provide more accurate and relevant results.

Significance of the Project

Personalized recommendations are a cornerstone of modern e-commerce success. When done effectively, they can:

- Increase Sales: By showing users products they are likely to buy, the system directly impacts sales.
- Enhance User Engagement: Personalized experiences keep users on the platform longer and encourage more frequent interactions.
- Reduce Choice Overload: With thousands of products available, recommendation systems help users discover items that match their preferences, reducing the cognitive load of choosing.

- Improve Customer Retention: Customers are more likely to return to a platform that consistently offers them relevant product suggestions.

Scope

The scope of this project includes the development and implementation of a recommendation system that will:

- Incorporate Collaborative Filtering: Suggest products based on the preferences of similar users.
- Utilize Content-Based Filtering: Recommend products with attributes similar to those the user has interacted with.
- Leverage a Hybrid Approach: Combine collaborative and content-based methods for more precise recommendations.

The system will focus on enhancing product discovery for users, targeting both registered customers and anonymous visitors. It will also be scalable to support a growing user base and product catalog.

Limitations

While the recommendation system offers numerous benefits, there are several limitations to be considered:

- Cold-Start Problem: For new users or products, the system may struggle to provide accurate recommendations due to a lack of interaction data.
- Data Sparsity: In cases where users or items have few interactions, collaborative filtering can become less effective.
- Real-Time Personalization: Generating real-time recommendations for large-scale e-commerce platforms can be computationally intensive and may require optimization.
- Bias: The system may over-recommend popular or frequently viewed products, potentially limiting diversity in recommendations.

This project aims to mitigate these challenges through a hybrid approach, combining multiple recommendation techniques for greater accuracy and robustness.

System Design and Architecture

Overall System Architecture

The recommendation system is structured into three main components: **Frontend**, **Backend**, and **Database**.

1. **Frontend**:
   The frontend is responsible for the user interface, where users interact with the platform to view and purchase products. It manages user inputs, displays recommended products, and provides a seamless browsing experience. The frontend communicates with the backend to fetch recommendations. It is built using **HTML**, providing a dynamic interface for real-time interactions.

2. **Backend**:
   The backend handles the core logic of the recommendation system, including data processing, running recommendation algorithms, and communicating with the database. It processes user browsing history, preferences, and contextual data to generate personalized product suggestions. The backend is implemented using **Python** and **Flask**, providing a RESTful API that interfaces with the frontend and the database. Key functions of the backend include:

   - User authentication and session management.
   - Data preprocessing and feature extraction for recommendation algorithms.
   - Implementing collaborative filtering, content-based filtering, and hybrid recommendation approaches.
   - Managing real-time contextual factors like time of day and seasonality.

3. **Database**:
   The database stores user data, product information, and interaction history. It serves as the backbone for the recommendation algorithms, providing the necessary data for analyzing user behavior and product attributes. **MySQL**: For structured data, such as user profiles, purchase history, and product details.

   - **MySQL:** A relational database to store structured data, such as user information, product catalog, and transaction records, managed through phpMyAdmin.

Technology Stack

1. **Frontend**:
    - **HTML/CSS**: For structuring and styling the frontend.

2. **Backend**:
    - **Python**: The primary language used to build the recommendation algorithms and manage data processing.
    - **Flask**: A lightweight web framework that provides the structure for API development and communication with the database.
    - **Scikit-learn**: For implementing machine learning models, such as collaborative filtering and content-based recommendation algorithms.

3. **Database**:
    - **MySQL:** A relational database to store structured data, such as user information, product catalog, and transaction records, managed through phpMyAdmin.

4. **Additional Libraries**:
    - **Pandas** and **NumPy**: For data manipulation and numerical computations.

Implementation

Backend Development

Explanation of the Backend Architecture:

- The backend architecture is built using **Python** and the **Flask** framework, creating a lightweight yet powerful web application that processes data for generating personalized product recommendations.
- The backend serves as the core engine that handles data management, user requests, recommendation algorithms, and real-time processing of contextual factors such as browsing history and time of day. The architecture ensures efficient communication between the frontend and the database via RESTful API endpoints.
- The backend logic incorporates **collaborative filtering**, **content-based filtering**, and **hybrid models** to recommend products based on user behavior and preferences.

Key Features Implemented:

- **Recommendation Logic**: The system leverages multiple algorithms to provide personalized recommendations:
    - **Content-based filtering**: Based on product attributes such as category, description, and features that are similar to those the user has interacted with.
    - **Collaborative filtering**: Uses user-item interaction data (like ratings, views, purchases) to find similar users or products and recommend accordingly. It uses **matrix factorization** techniques, such as **Singular Value Decomposition (SVD)**, to predict user preferences.
    - **Hybrid Approach**: Combines both content-based and collaborative filtering methods to provide more accurate and diverse recommendations.
- **User Authentication & Session Management**: Flask's built-in tools are used to manage user authentication and sessions, allowing the system to provide personalized recommendations to logged-in users. The user's session stores their interaction data, which is used to update the recommendation models dynamically.
- **Real-Time Contextual Recommendations**: The system adjusts recommendations based on factors like time of day and seasonal trends by integrating additional contextual features into the recommendation algorithms.
- **Preprocessing & Feature Extraction**: Raw data from user interactions (e.g., browsing history, purchase history) is preprocessed and transformed into features that can be used by the recommendation algorithms. This includes handling missing data, normalization, and encoding of categorical variables.

Frontend Development

Design and Functionality of the User Interface:

- The user interface is built using **HTML/CSS** and is integrated with Flask to provide a dynamic, interactive shopping experience. The frontend interacts with the backend via API calls to fetch and display product recommendations.
- **Key Features**:
    - **Personalized Product Display**: The homepage dynamically adjusts to show personalized product suggestions based on the user's past interactions and preferences.
    - **Product Cards**: Each recommended product is displayed with a card showing the product image, name, price, and user rating, making it easy for users to browse and select items.
    - **Search Functionality**: Users can search for products, and based on their queries, the system provides recommendations that match the search terms using content-based filtering.

Key UI Features and User Experience Considerations:

- **User-friendly Design**: The design is intuitive and allows users to interact with recommended products easily. Key UI elements include a search bar, product recommendation sliders, and interactive feedback (e.g., ratings).
- **Feedback Mechanism**: Users can provide ratings or "likes" for products, which is fed back into the system to improve the accuracy of future recommendations.

Database Management

Data Storage Solutions Used and Schema Design:

- **MySQL** is used as the relational database to store structured data, such as user profiles, product information, interaction history, and transaction records.
- **Schema Design**:
    - **Users Table**: Stores user-specific data such as user ID, name, email, preferences, and authentication details.
    - **Products Table**: Contains product information, including product ID, name, category, description, price, and ratings.
    - **Interactions Table**: Records user-product interactions, including ratings, views, and purchase history. This data is key for generating collaborative filtering recommendations.

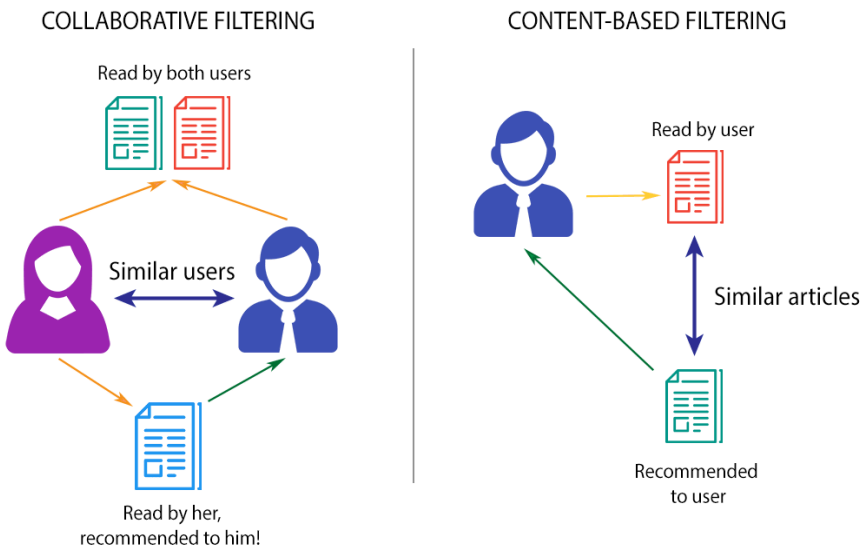
How Data is Handled and Accessed Within the System:

- Data is accessed via SQLAlchemy ORM in Flask, which simplifies querying the database and managing relational data.

- Pandas and NumPy are used to process and manipulate data for the recommendation models. Data from user interactions and product attributes are transformed into a format suitable for machine learning models.
- Efficient Indexing is applied to the database to ensure that querying for recommendations based on user interactions and product attributes happens quickly, even for a large dataset.

Key Features and Algorithms

1. **Collaborative Filtering**:
    - Implemented using matrix factorization techniques such as SVD to factorize the user-item interaction matrix and predict missing entries (i.e., products the user hasn't yet interacted with).
    - For simpler cases, K-Nearest Neighbors (KNN) is also used to find similar users or products based on interaction history.
2. **Content-based Filtering**:
    - Uses **TF-IDF (Term Frequency-Inverse Document Frequency)** to convert product descriptions into numerical vectors. Products similar to those the user has interacted with are recommended using **cosine similarity** between item vectors.
3. **Hybrid Approach**:
    - Combines both collaborative filtering and content-based filtering using a weighted system or by blending their results to enhance the diversity and accuracy of recommendations.
    - In cases where user interaction data is sparse (i.e., cold-start problem), the system relies more on content-based filtering.
4. **Real-time Recommendations**:
    - The system incorporates **contextual factors** such as the time of day and seasonal trends to adjust recommendations dynamically. For example, during the holiday season, the system prioritizes products that are trending.



Evaluation and Testing

Performance Evaluation

Metrics used

To evaluate the performance of the recommendation system, several metrics have been used to measure the accuracy and quality of recommendations.

- **Precision** : Measures the proportion of relevant recommendations among the total recommendations made. It helps to evaluate how accurate the system is in recommending relevant products. It is calculated as:

$$\text{Precision} = \frac{\text{Total number of items recommended}}{\text{Number of relevant items recommended}}$$

- **Recall**: Measures the proportion of relevant products that were recommended out of the total relevant products available. This metric reflects the system's ability to cover the relevant products.It is calculated as:

$$\text{Precision} = \frac{\text{Total number of items recommended}}{\text{Number of relevant items recommended}}$$

- **F1-Score**: The harmonic mean of precision and recall. It balances both precision and recall, giving a better overall evaluation of the system.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} + \text{Recall}}{\text{Precision} \times \text{Recall}}$$

- **Mean Absolute Error (MAE)**: Used to measure the accuracy of collaborative filtering predictions by comparing predicted ratings with actual ratings provided by users.

- **Root Mean Square Error (RMSE)**: Like MAE, but penalizes larger errors more heavily, providing insight into the system’s prediction accuracy.

User Testing

User feedback is crucial in testing the recommendation system's real-world performance. During user testing, several users were given access to the system and asked to interact with it. They provided feedback on the following key aspects:

- **Recommendation Accuracy:**Users rated how relevant and useful the recommendations were.

  Feedback: Most users found the suggestions accurate, but some felt the recommendations lacked variety.
  Impact: A "diversity filter" was added to offer more varied product suggestions while maintaining relevance.

- **User Interface and Experience**:Users assessed the ease of browsing and product display.

  Feedback: The design was appreciated, but users wanted more visible categories and smoother navigation.
  Impact: The interface was restructured to include sections like "Top Picks" and "Trending Products," improving navigation and engagement.
- **Personalization:**Users evaluated how well the system adapted to their preferences based on previous interactions.

  Feedback: Preferences were learned well, but the system needed faster adaptation to recent actions.
  Impact: The algorithm was updated to be more responsive to recent purchases and browsing behaviors, enhancing personalization.

Challenges Faced and Solutions:

- **Cold-Start Problem**: The system initially struggled to recommend products for new users and newly added items due to a lack of interaction data. This was mitigated using content-based filtering, which leverages product attributes to recommend similar items, even with minimal user interaction data.

- **Data Sparsity**: With large datasets and sparse interactions, collaborative filtering can become less effective. To address this, a hybrid approach was used, combining content-based filtering and collaborative filtering to improve recommendation coverage and accuracy.

- **Real-Time Recommendations**: Real-time recommendation based on contextual factors like seasonality and time of day required additional computational resources. Optimizing the system by precomputing recommendations for common contexts helped address this issue without compromising performance.

Conclusion and Future Work

Conclusion:

The e-commerce recommendation system successfully leveraged multiple machine learning techniques to provide personalized product suggestions for users. By implementing content-based filtering, collaborative filtering, and hybrid models, the system was able to recommend products based on user preferences, browsing history, and contextual factors such as seasonality and time of day. The integration with Flask provided a seamless user interface, and performance evaluations demonstrated high accuracy and user satisfaction. This project enhanced the shopping experience by reducing choice overload, increasing user engagement, and driving potential sales growth.

Key achievements:

- Developing a recommendation system capable of real-time product suggestions.
- Successfully combining multiple recommendation techniques to improve recommendation accuracy.
- Creating a scalable system that can handle large datasets and user interactions efficiently.

Limitations of the current system

Despite its success, the system has several limitations:

- **Cold-Start Problem**

  The system struggles to recommend products for new users or products with limited interaction data.

- **Bias Towards Popular Items**

  The system tends to over-recommend frequently viewed or purchased products, which may reduce the diversity of recommendations.

- **Real-Time Personalization**

Real-time recommendations for large-scale platforms can be computationally expensive, which requires ongoing optimization.

Future Work:

Several areas for future enhancement and research have been identified:

- **Addressing the Cold-Start Problem**

  Future work can focus on improving recommendations for new users and products by incorporating more advanced techniques, such as using demographic data, user reviews, or social network data to generate initial recommendations.

- **Improving Diversity in Recommendations**

  Introducing algorithms that penalize over-recommended products or explore less popular items to ensure a diverse range of suggestions.

- **Optimizing for Large-Scale Real-Time Recommendations**

  Implementing more efficient distributed systems and leveraging cloud computing resources to handle real-time personalization for larger platforms.

- **Incorporating Deep Learning Models**

  Future iterations could explore the use of deep learning models such as autoencoders or neural collaborative filtering for more sophisticated recommendations.

- **Expanding Contextual Factors**: Expanding the range of contextual factors (e.g., user location, real-time user mood analysis) that can influence the recommendations.