23

Building a Recommendation Engine

Recommended for you

23.1
Recommendation Engines.

What are Recommendation Engines

- Recommendation engines use algorithms to suggest relevant items to users by analyzing their past behavior, and for new users, they may recommend best-selling or high-profit products
- Implementing personalized recommendations based on user needs and interests enhances user experience and encourages repeat visits

What are Recommendation Engines

- The core challenge is providing recommendations that align with individual user interests, rather than generic suggestions based on majority preferences
- To address this, advanced recommendation engines use personalized methods for more accurate product suggestions, which we will explore in the following steps

23.2
Data Handling.

Data Collection Types

- Data can be collected by two means:
  - Explicit: These are pieces of information that are provided intentionally. Such as a rating, like, dislike, share, etc.
  - Implicit: These data points are gathered from data streams such as clicks, order history, and search history

Data Storage

- Both implicit and explicit data must be stored in databases
- Most standard approach for this is to use an SQL database
- The database will be used to train models

23.3
Types of **Filtering**.

Content-based Filtering

- Content-based filtering is a method used by recommendation systems to suggest items to users based on their individual preferences and interests
- **User**: The system starts by creating a user profile that captures the user's preferences and interests. This profile is built by analyzing various factors such as the user's past behavior, items they have liked, rated, or purchased

Content-based Filtering

- **Item**: Each item in the system's database is described using a set of features. In the case of movies, these features could include genre, director, cast, release year, etc.
- This description forms an 'item profile'

Content-based Filtering

- The core of content-based filtering is to match the user profile with the item profiles.
- This is usually done through techniques like calculating the similarity between the user's preferences and the item's characteristics.
- Common methods for this include calculating cosine similarity, Euclidean distance, or Pearson correlation

Content-based Filtering

- Based on the similarity scores, the system recommends items that closely match the user's profile
- For instance, if a user has shown a preference for science fiction movies, the system will primarily recommend movies from this genre
- The system can refine its recommendations over time by incorporating user feedback

Content-based Filtering

- One of the main advantages of content-based filtering is its ability to provide personalized recommendations
- However, it also has limitations:
  - potentially narrow range of suggestions
  - the system might struggle to provide accurate recommendations for new users
  - Might over specialize providing users content

Content-based Filtering

- One of the main advantages of content-based filtering is its ability to provide personalized recommendations
- However, it also has limitations:
  - potentially narrow range of suggestions
  - the system might struggle to provide accurate recommendations for new users
  - Might over specialize providing users content

Collaborative Filtering (User-User)

- User-user collaborative filtering emphasizes finding similarities between users based on their behavior or preferences
- It operates on the principle that if User A and User B have rated or liked similar items in the past, they will likely have similar preferences in the future
- This system also creates a unique user profile based on their tracked behaviors

Collaborative Filtering (User-User)

- The algorithm calculates similarity scores between users. This is done by comparing the user profiles, often using methods like Pearson correlation or cosine similarity
- Once similar users are identified, the system recommends items that have been liked or rated highly by these similar users but not yet experienced by the target user
- This system also learns and improves dynamically

Collaborative Filtering (User-User)

- While user-user collaborative filtering excels in creating a personalized experience by leveraging the power of community preferences, it can face challenges like scalability
- As the number of users grows, the computation of similarities becomes more complex
- It also has a cold start problem

Collaborative Filtering (Item-Item)

- Unlike user-user collaborative filtering, which focuses on finding similar users, item-item collaborative filtering concentrates on identifying similarities between items
- It's based on the premise that if a user likes a particular item, they are likely to enjoy items that are similar to it

Collaborative Filtering (Item-Item)

- Once similar items are identified, the system recommends these items to users who liked the initial item
- For instance, if a user liked Movie A, and Movie A is similar to Movies B and C, then Movies B and C will be recommended to this user

Collaborative Filtering (Item-Item)

- Item-item collaborative filtering often scales better than user-user collaborative filtering as the number of items in most systems is usually lower and grows more slowly than the number of users
- Additionally, item preferences tend to change less frequently than user preferences, providing more stability to the recommendation model

23.4
Matrix Factorization Technique.

Why Matrix Factorization

- Matrix factorization is a powerful technique used in recommendation engines, particularly for collaborative filtering
- It addresses some of the limitations of basic collaborative filtering by dealing with scalability and sparsity of user-item matrices

User-Item Matrix

- In a typical recommendation system, you start with a user-item matrix where rows represent users, columns represent items, and the values in the matrix represent user interactions with items
- However, this matrix is usually sparse, meaning most of the values are unknown or zero, as not every user interacts with every item

Goal of Factorization

- The goal of matrix factorization is to approximate this sparse user-item matrix by factorizing it into two lower-dimensional matrices whose product is a close approximation to the original matrix
- These two matrices typically represent 'latent features' of users and items

Goal of Factorization

- The idea is that both items and users can be described by a small set of underlying (latent) features
- For example, in a movie recommendation system, these latent features might include genres, movie length, or director style for movies, and preferences for these aspects for users

Decomposition Process

- Techniques like Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF), or Alternating Least Squares (ALS) are used to decompose the original matrix into user and item matrices
- Each row of the user matrix corresponds to a user's affinity to the latent features, and each column of the item matrix corresponds to an item's relation to these features

Predicting Missing Values

- By multiplying the user and item matrices, you can predict the missing values in the original matrix
- These predicted values represent the user's likely preference for items they haven't interacted with
- The system can then recommend items to a user based on these predicted preferences, typically suggesting items with the highest predicted ratings

Advantages and Challenges

- Matrix factorization is effective at discovering the underlying structures in the user-item matrix, handling sparsity and scalability issues, and providing more accurate recommendations
- However, this method has challenges like dealing with the cold start problem (new users or items with little to no data), and it requires choosing the right number of latent features to balance between overfitting and underfitting

23.5
Evaluation Metrics for Recommendation Engines.

Recall

- This measures the proportion of items that a user likes and that are actually recommended by the system
- It's calculated as the number of liked and recommended items (true positives) divided by the total number of items the user likes (true positives plus false negatives)
- For instance, if a user likes 5 items and the system recommends 3 of these, the recall is 60%. A higher recall indicates better recommendations

Precision

- This metric assesses the accuracy of the recommendations by calculating the proportion of recommended items that the user actually likes
- It's the number of liked and recommended items (true positives) divided by the total number of recommended items (true positives plus false positives)
- If out of 5 recommendations a user likes 4, the precision is 80%. Higher precision signifies better quality recommendations

MAP at k (Mean Average Precision at cutoff k)

- Unlike precision, MAP at k considers the order of recommendations
- It calculates precision at each point in the recommendation list and averages these values
- For example, for recommendations [0, 1, 1], the MAP at k would be 0.38, indicating the average precision across the list

NDCG (Normalized Discounted Cumulative Gain)

- This metric differs from MAP as it assigns a relevance score to each item, rather than treating recommendations as simply correct or incorrect
- In the case of recommending 4 out of 5 relevant movies (A to D), the NDCG would be 1, indicating perfect relevance of the recommendations.

END.