Event Recommendation System

Introduction

The proliferation of online events and platforms has made personalized recommendations crucial for improving user experience. This work focuses on a recommendation system that initially relies on content-based filtering to recommend events by analyzing users' historical domain preferences and continuously update the user profile based on new interactions, real-time click and like data

Problem Statement

Users often face difficulty in selecting suitable events from a vast pool of options. Our goal is to design a system that recommends events aligned with a user’s past interests, while enabling future expansion toward real-time personalization and collaborative filtering.

Entities

User : Input from users database

Events : Output from events database

Approaches

Batch Model (why not)

We can't use batch or offline model directly in this scenario because-

- ➔ At starting we don't have events (or too many events to train and recommend)
- ➔ the event (output) dataset is not static, new events keeps adding in the dataset
- ➔ the human behaviour is dynamic (need to monitor regularly)

Using Scheduler (Plan A)

- Use batch model
- Fetch data & train model regularly manually using scheduler

Online Model (Plan B)

- So we have to use **online (Incremental) model**
- It's a dynamic model which will train regularly in a fixed duration
- Model will change as new data arrives
- More uses more accurate results
- Algorithms used - incremental Stochastic Gradient Descent (SGD), Passive-Aggressive algorithms, Perceptron and Adaline.

Jugaad (Plan C)

- Combination of plan A and B
- Use partial_fit() to train on only new data regularly
- Then use scheduler for regular training

Data Needed

- The most liked domain of the user (taken as an input at the time of user registration)
- Last 10 previous search or explored history (from user database that will stores this information)
- Attended events (from user database)
- Wishlist or liked events (optional)

Methodology

❖ Data Representation

1. User Profile Creation

- Extract features from past events liked or attended by the use
- Domains such as: `Technology`, `Music`, `Sports`, `Art`, `Finance`, etc.
- Build a user preference vector from domain frequencies or TF-IDF-style weights.

2. Event Feature Extraction

- Represent each event with its content features (metadata like - category, tags, keywords).
- Apply vectorization techniques (TF-IDF, Bag of Words, embeddings).

❖ **Recommendation Process**

1. Similarity Calculation

- Use Cosine Similarity between the user profile vector and event feature vectors.
- Rank events based on similarity scores.

2. Recommendation Generation

- Recommend top-N events most similar to the user's interest vector

Technologies Used

- Python
- Pandas, NumPy
- Scikit-learn
- Matplotlib/Seaborn
- NLP
- Cosine Similarity
- Collaborative filtering

Future Work

- **Add User-Based Collaborative Filtering**: Leverage behavior of similar users for improved recommendation diversity.

- **Hybrid Model**: Combine content-based and user-based collaborative methods for robustness.