Video Recommendation Algorithm

Rajnandani Ambasana
B. Tech – Maths and Computation (DAIICT)
Email- rajnandania2004@gmail.com
No. : 7567760175

Introduction

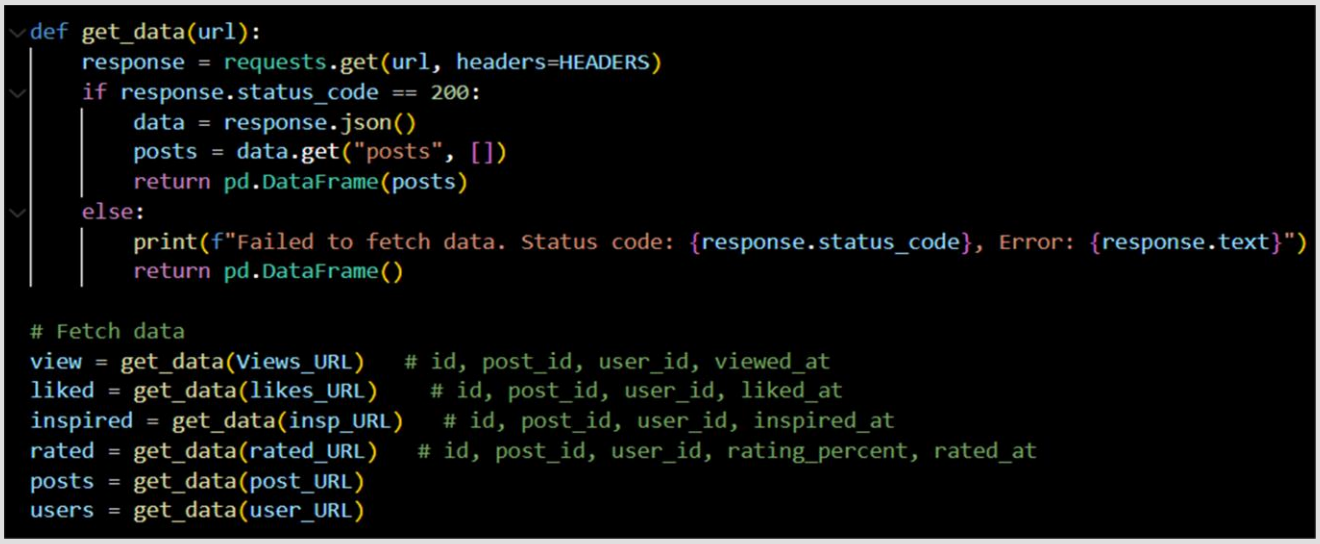
A recommendation system is a tool designed to predict what a user might like based on their past behavior and preferences. There are three main types of recommendation systems:

1. **Content-Based Filtering:** Recommends items similar to what the user liked before.
2. **Collaborative Filtering:** Recommends items based on what similar users liked.
3. **Hybrid Models:** Combines content-based and collaborative filtering for better accuracy.

Here I have created the algorithm based on all these three approaches. So let's see the algorithms.

Code structure

1. **Data Collection:**
   Data is fetched from APIs for user interactions (views, likes, inspirations, ratings) and metadata (posts, users). This data is merged into a single dataset to capture user-post interactions.



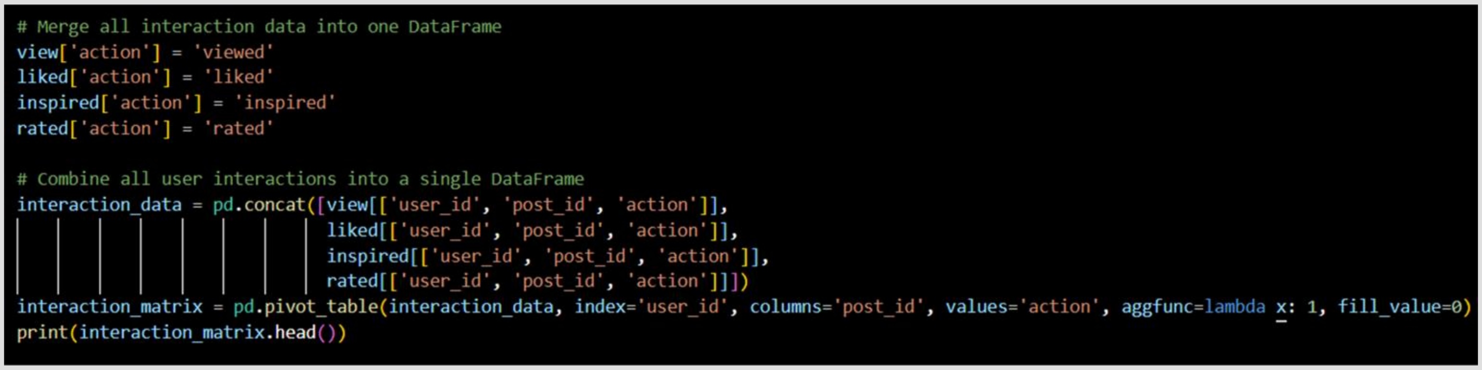
```python
def get_data(url):
    response = requests.get(url, headers=HEADERS)
    if response.status_code == 200:
        data = response.json()
        posts = data.get("posts", [])
        return pd.DataFrame(posts)
    else:
        print(f"Failed to fetch data. Status code: {response.status_code}, Error: {response.text}")
        return pd.DataFrame()

# Fetch data
view = get_data(Views_URL)   # id, post_id, user_id, viewed_at
liked = get_data(likes_URL)    # id, post_id, user_id, liked_at
inspired = get_data(insp_URL)   # id, post_id, user_id, inspired_at
rated = get_data(rated_URL)   # id, post_id, user_id, rating_percent, rated_at
posts = get_data(post_URL)
users = get_data(user_URL)
```

Tools Used:

- **requests:** To make HTTP requests and fetch JSON data.
- **pandas:** Converts JSON into a structured DataFrame for easy manipulation.

2. Interaction Matrix:

A pivot table is created with users as rows, posts as columns, and interactions as binary values. This matrix serves as the foundation for similarity calculations.



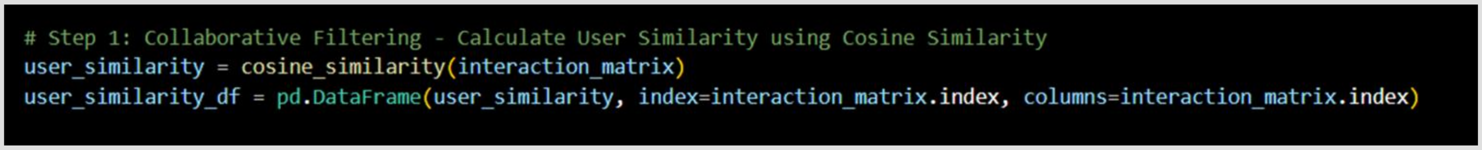
```python
# Merge all interaction data into one DataFrame
view['action'] = 'viewed'
liked['action'] = 'liked'
inspired['action'] = 'inspired'
rated['action'] = 'rated'

# Combine all user interactions into a single DataFrame
interaction_data = pd.concat([view[['user_id', 'post_id', 'action']],
                              liked[['user_id', 'post_id', 'action']],
                              inspired[['user_id', 'post_id', 'action']],
                              rated[['user_id', 'post_id', 'action']]])
interaction_matrix = pd.pivot_table(interaction_data, index='user_id', columns='post_id', values='action', aggfunc=lambda x: 1, fill_value=0)
print(interaction_matrix.head())
```

Tools used:

- **concat:** Joins multiple DataFrames.
- **pivot_table:** Converts the data into a user-item interaction matrix where rows are users and columns are posts. Values are 1 if an interaction exists.

3. Collaborative Filtering:

User similarity is computed using cosine similarity, which measures how closely users' interactions align. Posts liked by similar users but not yet interacted with by the target user are recommended.

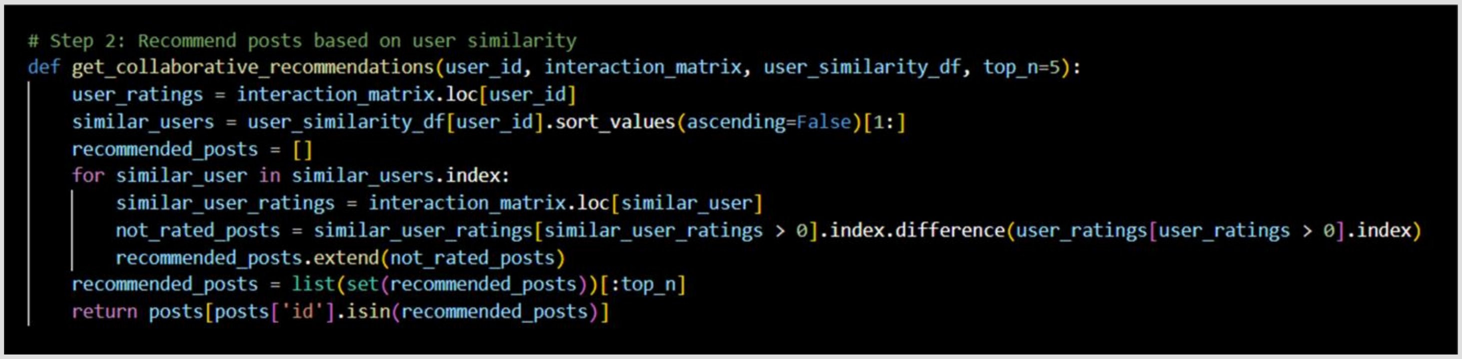


```python
# Step 1: Collaborative Filtering - Calculate User Similarity using Cosine Similarity
user_similarity = cosine_similarity(interaction_matrix)
user_similarity_df = pd.DataFrame(user_similarity, index=interaction_matrix.index, columns=interaction_matrix.index)
```

Tools used:

- **cosine_similarity:** Measures similarity between users by comparing their interaction vectors. High similarity indicates users have interacted with similar posts.

Recommendation Logic:

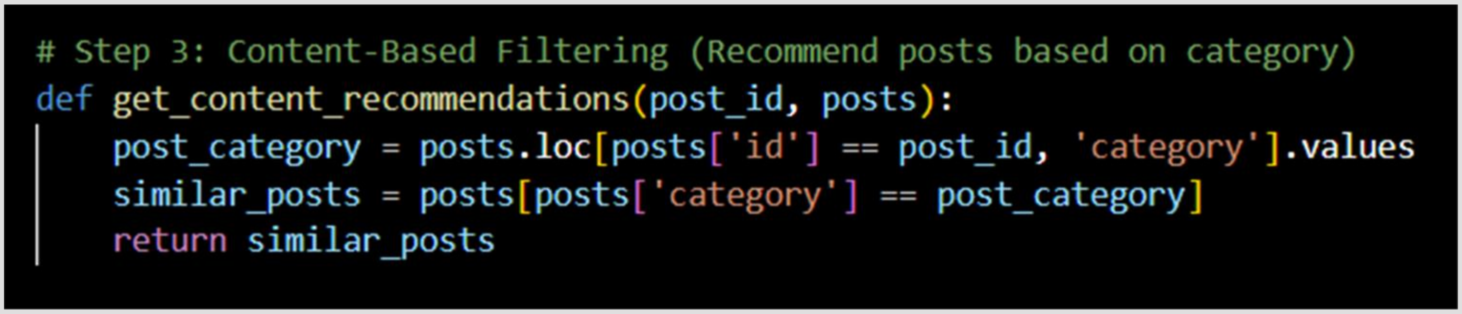


```python
# Step 2: Recommend posts based on user similarity
def get_collaborative_recommendations(user_id, interaction_matrix, user_similarity_df, top_n=5):
    user_ratings = interaction_matrix.loc[user_id]
    similar_users = user_similarity_df[user_id].sort_values(ascending=False)[1:]
    recommended_posts = []
    for similar_user in similar_users.index:
        similar_user_ratings = interaction_matrix.loc[similar_user]
        not_rated_posts = similar_user_ratings[similar_user_ratings > 0].index.difference(user_ratings[user_ratings > 0].index)
        recommended_posts.extend(not_rated_posts)
    recommended_posts = list(set(recommended_posts))[:top_n]
    return posts[posts['id'].isin(recommended_posts)]
```

- **Logic:** Find posts liked by similar users that the current user hasn't interacted with.
- **Tools:** Filtering and ranking based on user similarity.

4. Content-Based Filtering:

Posts are recommended based on shared attributes (e.g., category). For a given post, other posts with the same category are suggested.

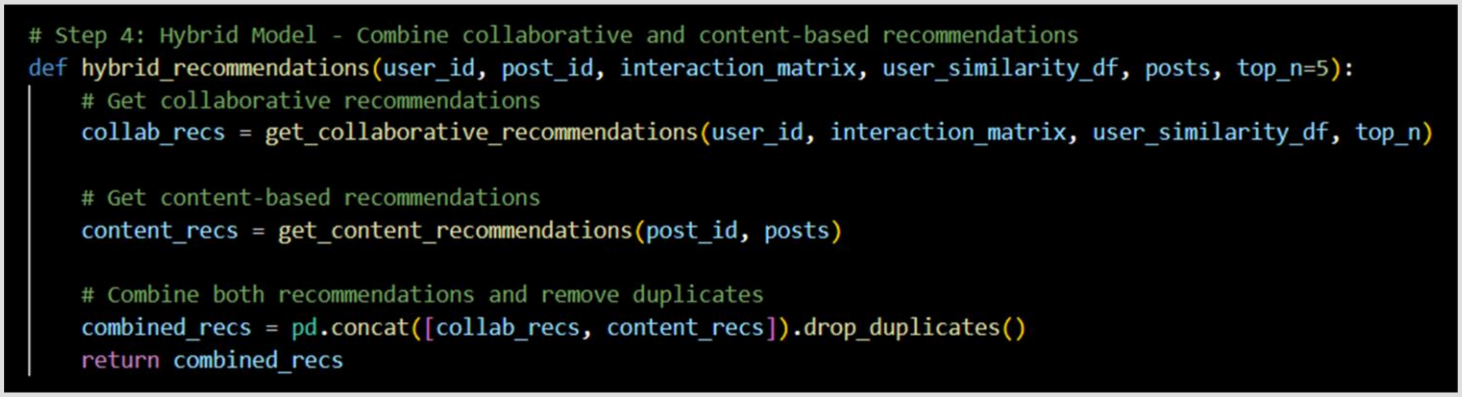


```python
# Step 3: Content-Based Filtering (Recommend posts based on category)
def get_content_recommendations(post_id, posts):
    post_category = posts.loc[posts['id'] == post_id, 'category'].values
    similar_posts = posts[posts['category'] == post_category]
    return similar_posts
```

Purpose: Recommend posts from the same category as the given post.
Logic: Matches categories of posts using a simple filter.

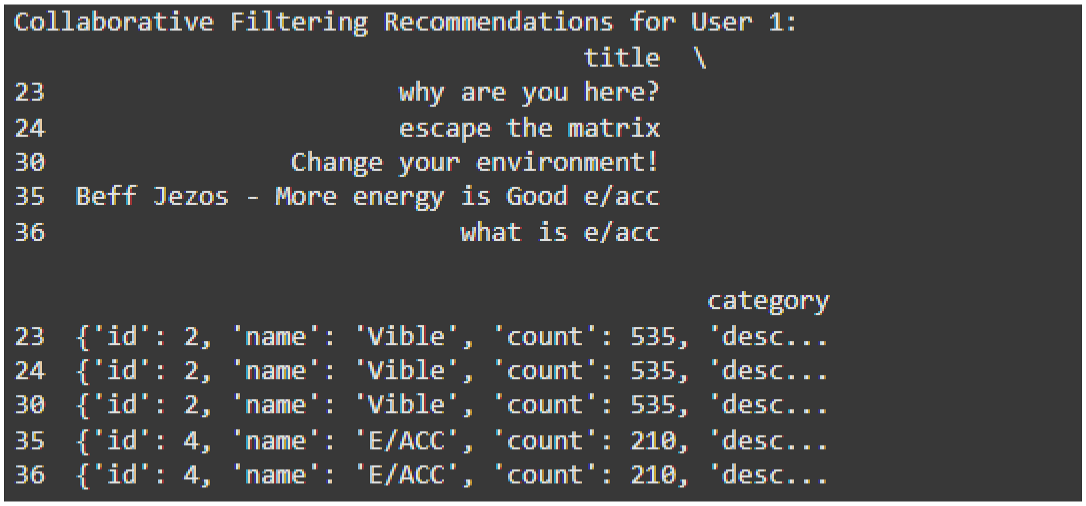
5. Hybrid Recommendation Model:

Combines collaborative and content-based recommendations to provide diverse and personalized suggestions. Results from both models are merged, ensuring uniqueness.



```python
# Step 4: Hybrid Model - Combine collaborative and content-based recommendations
def hybrid_recommendations(user_id, post_id, interaction_matrix, user_similarity_df, posts, top_n=5):
    # Get collaborative recommendations
    collab_recs = get_collaborative_recommendations(user_id, interaction_matrix, user_similarity_df, top_n)

    # Get content-based recommendations
    content_recs = get_content_recommendations(post_id, posts)

    # Combine both recommendations and remove duplicates
    combined_recs = pd.concat([collab_recs, content_recs]).drop_duplicates()
    return combined_recs
```

Output


```
Collaborative Filtering Recommendations for User 1:
                                     title  \
23                       why are you here?
24                       escape the matrix
30                Change your environment!
35  Beff Jezos - More energy is Good e/acc
36                           what is e/acc

                                             category
23  {'id': 2, 'name': 'Vible', 'count': 535, 'desc...
24  {'id': 2, 'name': 'Vible', 'count': 535, 'desc...
30  {'id': 2, 'name': 'Vible', 'count': 535, 'desc...
35  {'id': 4, 'name': 'E/ACC', 'count': 210, 'desc...
36  {'id': 4, 'name': 'E/ACC', 'count': 210, 'desc...
```

Conclusion

This recommendation system uses collaborative and content-based filtering to make personalized suggestions. It looks at how users interact with posts and the type of content in the posts to give better recommendations. By combining both methods, the system provides useful and varied suggestions, making it easier for users to find content they like. This approach shows how data can improve user experience on online platforms.

I sincerely thank Persist Ventures for providing me with the opportunity to work on this assignment. It is an honor to showcase my skills and dedication through this project. I deeply appreciate your consideration and look forward to the possibility of contributing further to your esteemed organization.