Mood-Based Video Recommendation System Using FastAPI and Hybrid Recommendation Algorithms

1. Project Overview

This project implements a **mood-based video recommendation system** for the web server using FastAPI. It combines **content-based filtering (CBF)**, **collaborative filtering (CF)**, and a **hybrid recommendation** approach to suggest relevant videos. The system uses **machine learning models**, data manipulation with **pandas**, and **TF-IDF** for feature extraction, addressing challenges like mood extraction from text summaries and resource optimisation for edge environments.

2. Technical Stack

- **FastAPI:** Web framework used for building the API and handling HTTP requests.
- **Pydantic:** This is used for data validation and request/response parsing.
- **Pandas:** Data manipulation and analysis library.
- **sklearn:** For machine learning techniques like **TF-IDF**, **SVD (Singular Value Decomposition)**, and **cosine similarity**.
- **Custom Algorithms:** Content-Based Filtering (CBF), Collaborative Filtering (CF), and a Hybrid recommendation approach. These are given in the algorithms.py file.
- **mood extractor:** A custom module for extracting moods from text data. There is a dictionary named emotions for every datapoint. From that I extracted the moods for each post. This function comes from the mood_extractor.py file.

3. System Architecture

The system is structured to handle video recommendation requests based on user input, considering factors like mood, category, and user interaction history. It utilizes the following components:

1. **Data Processing:**
    - This is one of the most important steps because, after careful observation, I found out that our data in JSON format contains various important features such as category and mood or emotions in some places, which were crucial for improving the model.
    - The data is loaded from `data_summary.json` and converted into a pandas data frame.
    - **Moods** are extracted from the video post summaries using the `extract_moods` function, followed by **mood counting** and filtering (only retaining moods with occurrences greater than 100).
2. **Recommendation Algorithms:**
    - **Collaborative Filtering (CF):** Uses **matrix factorization** with **Singular Value Decomposition (SVD)** to generate latent factors for users and items. The similarity between users is calculated using **cosine similarity**.
    - **Content-Based Filtering (CBF):** Uses the **TF-IDF vectorizer** to represent the content of the posts (moods and categories) as vectors. Cosine similarity is used to recommend videos based on the user's mood and category preference.
    - **Hybrid Approach:** Combines **CF** and **CBF** scores using a weighted average to provide the final recommendations, taking into account factors like upvotes, view counts, and rating counts to rank the recommendations.
3. **FastAPI Interface:**
    - **GET `/`:** Serves a simple HTML form where users can input their **username**, **mood**, and **category**.
    - **POST `/recommendations/`:** Accepts user input and uses the recommendation algorithms to return the top 10 video recommendations.

4. **Data Features:**
    - The system processes several features, including the **mood** of each post, the **category** of the video, and the user's interactions (e.g., upvotes).
    - **Interaction matrix:** The matrix is created by pivoting the data on user interaction counts (upvotes) and using this for collaborative filtering.

---

4. Machine Learning and Recommendation Algorithms

The recommendation logic relies on three primary techniques:

1. **Collaborative Filtering (CF):**
    - **Matrix Factorization:** The interaction matrix (user-video upvotes) is factorized using **SVD** to create a **latent matrix** of users and items. The cosine similarity between the user's latent vector and the latent vectors of other users is computed to predict preferences.
    - **Function:** `get_cf_scores(username, top_n=10)`
        - For a given user, the algorithm identifies the most similar users based on latent factors and recommends the videos interacted with by these users.
2. **Content-Based Filtering (CBF):**
    - **TF-IDF Vectorization:** The `moods` and `category_name` of the videos are combined into a feature vector for each video. Cosine similarity is then computed between the user's preference (based on mood and category) and the videos in the dataset.
    - **Function:** `get_cbf_scores(user_mood, user_category, top_n=10)`
        - This algorithm recommends videos that match the user's mood and category preferences based on the content of the posts.
3. **Hybrid Recommendation (CBF + CF):**
    - **Combining CF and CBF:** The hybrid method combines both CF and CBF recommendations, ranking the videos by a **hybrid score**, a weighted sum of factors such as **upvotes**, **view count**, and **rating count**.

- **Function:** `get_hybrid_recommendations(username, mood, category, top_n=10, alpha=0.7)`
    - The final recommendation set is created by merging the top CF and CBF recommendations and sorting by the **hybrid score.**

---

5. Data Flow and User Interaction

1. **User Input:**
    - The user enters their **username**, **mood**, and **category** via the web form. The available moods are predefined, and the category is selected from a list.
2. **Data Processing:**
    - **Moods Extraction:** The mood extraction function processes the video summaries and filters the moods based on their frequency.
    - **Interaction Matrix:** The system uses the interaction matrix to identify similar users and recommend videos based on prior interactions.
3. **Recommendation Generation:**
    - **Collaborative filtering** predicts videos based on user interaction patterns.
    - **Content-based filtering** recommends videos based on mood and category similarity.
    - **Hybrid approach** combines both techniques to provide the final recommendations.
4. **Output:**
    - The system returns the top recommended videos, including the **video title**, **ID**, and **video link**.

---

6. Challenges and Solutions

- **Handling Data Sparsity:** Collaborative filtering often struggles with sparse data. The hybrid approach helps mitigate this by incorporating content-based features, improving recommendation accuracy even with sparse user interactions.

- **Mood Classification:** Extracting and accurately classifying moods from text data can be challenging, especially with diverse user-generated content. This was managed using a custom `moodextractor` module, which leverages a pre-trained model to extract meaningful mood labels.
- **Performance Optimization:** The algorithms are designed to work efficiently on edge devices. Optimizations such as dimensionality reduction using SVD and feature extraction via TF-IDF help minimize computational costs for large datasets.

---

7. Conclusion

This project successfully combines **FastAPI** with **machine learning models** to build a mood-based video recommendation system. Integrating **collaborative filtering, content-based filtering, and a hybrid approach** provides personalized video recommendations tailored to the user's mood and category preferences. Future improvements will focus on optimizing the solution for edge devices and integrating more advanced AI features to enhance the recommendation experience.