Movie Recommender System Documentation

This document provides a detailed explanation of the code for a movie recommender system based on cosine similarity. The system predicts ratings for unrated movies and suggests top movies for individual users and overall recommendations. The system processes a ratings matrix stored in a CSV file.

---

1. Code Overview

The program is divided into the following components:
1. DataLoader Class: Loads and displays the ratings matrix from a CSV file.
2. SimilarityCalculator Class: Computes the cosine similarity between two users.
3. RecommenderSystem Class: Predicts movie ratings, suggests top recommendations for users, and calculates overall movie recommendations.
4. Main Function: Orchestrates the execution of the system.

---

2. Class Descriptions

2.1 DataLoader Class
Methods:
- `loadRatingsMatrix(const string &filePath)`
  - Reads a CSV file where each row represents a user and each column represents a movie.
  - Parses the data into a 2D vector of integers (ratings matrix).
  - Returns the ratings matrix.

- `printMatrix(const vector<vector<int>> &matrix)`
  - Prints the ratings matrix to the console.

Usage Example:
```cpp
string filePath = "ratings.csv";
vector<vector<int>> ratingsMatrix = DataLoader::loadRatingsMatrix(filePath);
DataLoader::printMatrix(ratingsMatrix);
```

2.2 SimilarityCalculator Class
Methods:
- `calculateCosineSimilarity(const vector<int> &user1, const vector<int> &user2)`

  - Calculates the cosine similarity between two users based on their ratings.
  - Formula:

$$
\text{Similarity} = \frac{\text{Dot Product of User Vectors}}{\text{Norm of User1} \times \text{Norm of User2}}
$$

  - Returns a value between 0 and 1, where higher values indicate greater similarity.

Usage Example:

```cpp
vector<int> user1 = {4, 0, 5, 3};
vector<int> user2 = {5, 1, 4, 0};
double similarity = SimilarityCalculator::calculateCosineSimilarity(user1, user2);
```

2.3 RecommenderSystem Class

Attributes:

- `ratingsMatrix`: Stores the ratings matrix loaded from the CSV file.

Methods:

- `predictRatings(int userIndex)`
  - Predicts ratings for unrated movies for a specific user.
  - For each unrated movie:
    - Computes the weighted sum of ratings from similar users.
    - Normalizes the sum using the total similarity of contributing users.
  - Returns a list of predicted ratings as pairs of movie indices and predicted values.

- `printPredictedRatings(int userIndex, const vector<pair<int, double>> &predictedRatings)`
  - Prints the predicted ratings for a given user.

- `suggestTopNMovies(int userIndex, const vector<pair<int, double>> &predictedRatings, int topN)`
  - Suggests the top `N` movies for a given user based on predicted ratings.

- `suggestTopNMoviesOverall(const vector<vector<pair<int, double>>> &allPredictedRatings, int topN)`
  - Aggregates predictions from all users to suggest the top `N` movies overall.

Usage Example:

```cpp
RecommenderSystem recommender(ratingsMatrix);
```

```cpp
vector<pair<int, double>> predictedRatings =
recommender.predictRatings(userIndex);
recommender.printPredictedRatings(userIndex, predictedRatings);
recommender.suggestTopNMovies(userIndex, predictedRatings, 3);
```

---

3. Main Function Workflow

1. Load the Ratings Matrix
   - Reads the ratings matrix from the file `ratings.csv` using `DataLoader`.
   - Displays the matrix.

2. Initialize Recommender System
   - Creates an instance of `RecommenderSystem` initialized with the ratings matrix.

3. Predict Ratings for Each User
   - Loops through all users and predicts ratings for their unrated movies.
   - Prints the predictions and suggests the top 3 movies for each user.

4. Suggest Top Movies Overall
   - Aggregates predicted ratings across all users.
   - Recommends the top 5 movies overall based on average predicted ratings.

Main Function Code:
```cpp
int main() {
    string filePath = "ratings.csv";

    vector<vector<int>> ratingsMatrix = DataLoader::loadRatingsMatrix(filePath);
    DataLoader::printMatrix(ratingsMatrix);

    RecommenderSystem recommender(ratingsMatrix);

    vector<vector<pair<int, double>>> allPredictedRatings;
    for (size_t userIndex = 0; userIndex < ratingsMatrix.size(); ++userIndex) {
        vector<pair<int, double>> predictedRatings =
recommender.predictRatings(userIndex);
        allPredictedRatings.push_back(predictedRatings);
        recommender.printPredictedRatings(userIndex, predictedRatings);
        recommender.suggestTopNMovies(userIndex, predictedRatings, 3);
        cout << "\n";
    }
    recommender.suggestTopNMoviesOverall(allPredictedRatings, 5);

    return 0;
}
```

---

4. File Structure

The program expects a CSV file with the following format:
- Rows represent users.
- Columns represent movies.
- Cell values are integers representing ratings (0 if the movie is unrated).

Example CSV Content:
```csv
4,0,5,3
0,2,3,5
1,0,0,4
```

---

5. Customization

- Top N Recommendations: Adjust the `topN` variable in `suggestTopNMovies` and `suggestTopNMoviesOverall` methods.
- File Path: Change the `filePath` variable in the main function to point to your ratings file.

---

6. Execution Steps

1. Prepare the ratings CSV file.
2. Compile and run the code using a C++ compiler:
   ```bash
   g++ -o recommender recommender.cpp
   ./recommender
   ```
3. View the predicted ratings and recommendations in the console output.
