```
import numpy as np
from collections import defaultdict

# Sample data: user ratings for movies (user_id: {movie_id: rating})
user_ratings = {
    1: {1: 5, 2: 3, 3: 4, 4: 3},
    2: {1: 3, 2: 1, 3: 2, 4: 3, 5: 4},
    3: {1: 4, 2: 3, 3: 4, 5: 5},
    4: {2: 4, 3: 3, 4: 5, 5: 2},
    5: {1: 3, 3: 5, 4: 4, 5: 1}
}

# Sample movie data
movies = {
    1: "The Shawshank Redemption",
    2: "The Godfather",
    3: "The Dark Knight",
    4: "Pulp Fiction",
    5: "Fight Club"
}

def get_movie_ratings(movie_id):
    return [(user, ratings[movie_id])
            for user, ratings in user_ratings.items()
            if movie_id in ratings]

def compute_similarity(movie1, movie2):
    ratings1 = get_movie_ratings(movie1)
    ratings2 = get_movie_ratings(movie2)

    users1 = set(user for user, _ in ratings1)
    users2 = set(user for user, _ in ratings2)
    common_users = users1 & users2

    if not common_users:
        return 0

    ratings1 = np.array([rating for user, rating in ratings1 if user in common_users])
    ratings2 = np.array([rating for user, rating in ratings2 if user in common_users])

    return np.corrcoef(ratings1, ratings2)[0, 1]

def get_similar_movies(target_movie, n=2):
    similarities = [(other_movie, compute_similarity(target_movie, other_movie))
                    for other_movie in movies if other_movie != target_movie]

    return sorted(similarities, key=lambda x: x[1], reverse=True)[:n]

def recommend_movies(user_id, n=2):
    user_movies = set(user_ratings[user_id].keys())
    all_movies = set(movies.keys())
    unwatched_movies = all_movies - user_movies

    movie_scores = defaultdict(float)

    for movie in user_movies:
        user_rating = user_ratings[user_id][movie]
        similar_movies = get_similar_movies(movie)

        for similar_movie, similarity in similar_movies:
            if similar_movie in unwatched_movies:
                movie_scores[similar_movie] += similarity * user_rating

    # Normalize scores
    for movie in movie_scores:
        movie_scores[movie] /= len(user_movies)

    top_recommendations = sorted(movie_scores.items(), key=lambda x: x[1], reverse=True)[:n]
    return [(movies[movie_id], score) for movie_id, score in top_recommendations]

# Example usage
if __name__ == "__main__":
    user_id = 1
    print(f"Top recommendations for User {user_id}:")
    recommendations = recommend_movies(user_id)
    for movie, score in recommendations:
        print(f"- {movie} (Score: {score:.2f})")

    # Let's also print some similar movies
    target_movie = 1
    print(f"\nMovies similar to {movies[target_movie]}:")
```

```
similar = get_similar_movies(target_movie)
for movie_id, similarity in similar:
    print(f"- {movies[movie_id]} (Similarity: {similarity:.2f})")
```

```
Top recommendations for User 1:
- Fight Club (Score: 0.87)

Movies similar to The Shawshank Redemption:
- The Godfather (Similarity: 0.87)
- Fight Club (Similarity: 0.69)
```