Hrutvi Barad

Movie Recommendation System Report

1. Introduction

Recommendation systems play a crucial role in improving user experience across platforms such as streaming services (Netflix, Spotify), e-commerce websites (Amazon), and social media (YouTube). They analyze user preferences and behaviors to suggest relevant items, enhancing engagement and satisfaction.

This project focuses on building a movie recommendation system using the MovieLens 100K dataset and explores three distinct approaches:

1. **User-Based Collaborative Filtering**: Recommends movies based on the preferences of similar users.

2. **Item-Based Collaborative Filtering**: Suggests movies similar to those a user has already rated.

3. **Random-Walk-Based Pixie Algorithm**: Simulates random walks on a bipartite graph to discover deeper connections between users and movies.

The goal is to evaluate the effectiveness of these methods and provide meaningful recommendations.

2. Dataset Description

The MovieLens 100K dataset, provided by GroupLens Research, is widely used for research in recommendation systems. It contains:

- **943 users**, each having rated at least 20 movies.
- **1682 movies**, spanning various genres.
- **100,000 ratings**, ranging from 1 to 5 stars.

Key features include:

- `user_id`: Unique identifier for each user.
- `movie_id`: Unique identifier for each movie.
- `rating`: The score assigned by a user to a movie.
- `timestamp`: The time when the rating was given.

Preprocessing Steps:

1. Converted timestamps into readable date formats using `pd.to_datetime()` for better interpretability.
2. Constructed matrices (e.g., User-Movie Matrix) for collaborative filtering techniques.
3. Built adjacency lists for graph-based methods, connecting users and movies based on ratings.

3. Methodology

This section describes the theoretical basis of the three approaches used in this project:

3.1 User-Based Collaborative Filtering

User-based collaborative filtering relies on identifying users with similar preferences based on their historical ratings. The key steps include:

- Calculating similarity between users using metrics such as cosine similarity or Pearson correlation.
- Identifying users most similar to the target user.
- Recommending movies that similar users have rated highly but the target user has not yet rated.

3.2 Item-Based Collaborative Filtering

Item-based collaborative filtering focuses on finding relationships between movies rather than users. The key steps include:

- Measuring similarity between movies based on shared user ratings using cosine similarity.
- Identifying movies that are most similar to those already rated by the target user.
- Recommending these similar movies to the user.

3.3 Random-Walk-Based Pixie Algorithm

The Pixie algorithm uses graph-based methods to simulate random walks over a bipartite graph of users and movies:

- Constructs a bipartite graph where nodes represent users and movies, and edges represent ratings.

- Performs weighted random walks starting from nodes (users or movies) to explore deeper connections in the graph.

- Movies visited frequently during walks are ranked higher for recommendation.

4. Implementation Details

4.1 Graph Construction

A bipartite graph was constructed using an adjacency list representation:

1. Each `user_id` node is connected to `movie_id` nodes representing the movies they have rated.

2. Each `movie_id` node is connected back to `user_id` nodes representing the users who rated them.

The adjacency list structure is implemented as a dictionary where:

- Keys are node IDs (`user_id` or `movie_id`).

- Values are sets of connected nodes.

Example adjacency list:

python

```
{
    196: {242, 302, 377},   # User 196 rated movies 242, 302, and 377
```

```
    242: {196, 186},        # Movie 242 was rated by users 196 and 186
}
```

4.2 Weighted Random Walks

Random walks were implemented as follows:

1. Start at a given node (`user_id` or `movie_id`).
2. At each step, transition to one of the neighboring nodes based on uniform probabilities or edge weights (ratings).
3. Record visited movie nodes during walks.

The algorithm ensures exploration of deeper connections in the graph while maintaining randomness in traversal.

Key parameters:

- `walk_length`: Number of steps in each random walk (default: 15).
- `num_walks`: Number of random walks performed for each starting node.

Example implementation:

```
def weighted_random_walk(graph, start_node, walk_length=15):
    visited_movies = []
    current_node = start_node

    for step in range(walk_length):
        neighbors = list(graph.get(current_node, []))
```

```
        if not neighbors:
            break
        next_node = random.choice(neighbors)
        if isinstance(next_node, int):  # Assuming movie_ids are
integers
            visited_movies.append(next_node)
        current_node = next_node

    return visited_movies
```

4.3 Collaborative Filtering

Both user-based and item-based collaborative filtering utilized cosine similarity matrices calculated using `sklearn.metrics.pairwise.cosine_similarity`.

Steps for User-Based Collaborative Filtering:

1. Constructed a User-Movie Matrix using Pandas' `pivot()` function:

   python

```
user_movie_matrix = ratings.pivot(index='user_id', columns='movie_id',
values='rating')
```

2.
3. Computed cosine similarity between rows (users):

   python

```
user_similarity = cosine_similarity(user_movie_matrix.fillna(0))
```

4.

5. Recommended top-rated movies from similar users.

Steps for Item-Based Collaborative Filtering:

1. Constructed a Movie-User Matrix (transpose of User-Movie Matrix).
2. Computed cosine similarity between rows (movies).
3. Recommended top-rated movies similar to those already rated by the target user.

5. Results and Evaluation

5.1 User-Based Collaborative Filtering

Example output:

```
Top 5 recommendations for User 196:
| Ranking | Movie Name                          |
|---------|-------------------------------------|
| 1       | Star Wars (1977)                     |
| 2       | Jurassic Park (1993)                |
| 3       | Independence Day (1996)             |
| 4       | Mission Impossible (1996)           |
| 5       | Toy Story (1995)                    |
```

5.2 Item-Based Collaborative Filtering

Example output:

```
Top 5 recommendations for Movie "Toy Story (1995)":
```

```
| Ranking | Movie Name                          |
|---------|-------------------------------------|
| 1       | GoldenEye (1995)                    |
| 2       | Four Rooms (1995)                   |
| 3       | Get Shorty (1995)                   |
| 4       | Copycat (1995)                      |
| 5       | Twelve Monkeys (1995)               |
```

5.3 Random-Walk-Based Pixie Algorithm

Example output:

```
Top 5 recommendations using Random Walks:
| Ranking | Movie Name                          |
|---------|-------------------------------------|
| 1       | The Godfather (1972)                |
| 2       | Pulp Fiction (1994)                 |
| 3       | Schindler's List (1993)             |
| 4       | Forrest Gump (1994)                 |
| 5       | Braveheart (1995)                   |
```

6. Conclusion

This project demonstrates the effectiveness of three distinct recommendation techniques:

1. **User-Based Collaborative Filtering** excels in identifying personalized recommendations based on shared preferences among users.

2. **Item-Based Collaborative Filtering** is ideal for suggesting content similar to what the user already enjoys.

3. The **Random-Walk-Based Pixie Algorithm** provides unique insights by leveraging graph traversal techniques.