GROUP SEQUENTIAL RECOMMENDATION

UJUNWA EDUM
ASHKAN KHADEMIAN

INTRODUCTION

Generally, the goal of sequential recommendation is to recommend items to users based on the knowledge of the past interaction. In order to achieve this both the group satisfaction and disagreement will be put into consideration.

The **sequential group recommendation implementation** proposed here offers a major benefit over the **hybrid sequential aggregation recommendation** methods stated in the class because it integrates dynamic metrics like satisfaction, disagreement, and diversity into its iterative process.

Improvement on Sequential hybrid aggregation model (proposed method)

This method will ensure fairness by reducing disagreement, satisfaction by maximizing preferences, and variety to enhance content diversity.

Some metrics used on implementation of our proposed method are

- preference score
- user satisfaction
- group satisfaction
- group disagreement
- genre diversity.

Implementation

```python
def generate_sequential_recommendations(
        group,
        user_ratings,
        movies,
        movie_genres,
        iterations=10,
        top_k=10,
        alpha=0,  # weight for least score
        beta=1,  # weight for average score
        gamma=0,  # weight for diversity score
):
```

```python
# Calculate scores for each movie
for movie in movies_iterator:
    avg_score = avgScore(group, movie, user_ratings)
    least_score = leastScore(group, movie, user_ratings)

    # Calculate temporary recommendations with this movie
    temp_recommendations = group_recommendations[-1][:top_k - 1] + [movie] if group_recommendations else [movie]
    diversity_score = calculate_genre_diversity(group, temp_recommendations, movie_genres, user_ratings)

    # Combine scores with weights
    movie_scores[movie] = (alpha * least_score +
                           beta * avg_score +
                           gamma * diversity_score)

# Select top movies based on combined score
top_movies = sorted(movie_scores, key=movie_scores.get, reverse=True)[:top_k]
group_recommendations.append(top_movies)

# Calculate metrics
group_sat = calculate_group_satisfaction(group, top_movies, user_ratings, user_satisfactions)
group_dis = calculate_group_disagreement(group, top_movies, user_ratings, user_satisfactions)
group_div = calculate_genre_diversity(group, top_movies, movie_genres, user_ratings)

# Update weights based on metrics with distinct strategies
# If satisfaction is low, increase beta (satisfaction weight)
if group_sat < 0.7:  # threshold for "low" satisfaction
    beta = min(0.6, beta + 0.05)  # increase but cap at 0.6

# If disagreement is high, increase alpha (fairness weight)
if group_dis > 0.20:  # threshold for "high" disagreement
    alpha = min(0.5, alpha + 0.05)  # increase but cap at 0.5

# If diversity is low, increase gamma (diversity weight)
if group_div < 0.9:  # threshold for "low" diversity
    gamma = min(0.4, gamma + 0.05)  # increase but cap at 0.4

# Normalize weights to sum to 1
total = alpha + beta + gamma
alpha = alpha / total
beta = beta / total
gamma = gamma / total
```

This main function uses three components to generate recommendations: average score (group preference), least score (fairness), and genre diversity (variety). it iterates over the specified number of rounds and updates the weights for the three components based on the group satisfaction, disagreement, and diversity scores. It prints the group recommendations for each iteration.

The three component scoring system

Movie Score= $\alpha \cdot leastScore + \beta \cdot avgScore + \gamma \cdot diversityScore$

Dynamic Weight Adjustment:

- α increases when disagreement is high
- β increases when satisfaction is low
- γ increases when diversity is low

Some core metrics used for this sequential recommendation method

User satisfaction
user_satisfaction = group_satisfaction / user_ideal_satisfaction

Group satisfaction
group_satisfaction = (1/|G|) * ∑(user_satisfaction(u) for u in group)

Group disagreement
```
disagreement = max(user_satisfactions) - min(user_satisfactions)
```

Genre Diversity
diversity = -∑(p_i * log(p_i)) / log(n_genres)