Assignment 4

Explanations to Why-Not questions in recommender systems

Harry & Kien

Atomic granularity

Why not Matrix?

Group granularity

Why not comedies?

Atomic position absenteeism

Why not rank Toy Story at the 2nd place?

Atomic granularity (1/2)

100 peers, 20 closest peers, numPI=5

```
reasons = []

if all_movies.loc[all_movies['movie title'] == movie_name].shape[0] == 0:
    reasons.append("The movie is not in the database")
    return reasons

# Find movies with a similar ratings in the group ratings
movie_id = all_movies.loc[all_movies['movie title'] == movie_name]['movie id'].iloc[0]
movie_rating = group_ratings.loc[group_ratings['movie id'] == movie_id]['average'].iloc[0]
same_rated_movies = group_ratings[group_ratings['average'] == movie_rating]

# Find the highest ranking among the similarly rated movies
# and determine if it ties with the movie in question
if same_rated_movies.shape[0]:
    first_same_rated_movie_id = same_rated_movies.iloc[0]['movie id']
    reindexed_group_ratings = group_ratings.reset_index(drop=True)
    same_rated_movie_rating_index = reindexed_group_ratings.loc[reindexed_group_ratings['movie id'] == first_same_rated_movie_id].index[0]
    if same_rated_movie_rating_index <= movie_num:
        reasons.append("This movie's predicted rating ties with many others who made the list")
        return reasons

# Find the ranking of the movie in question and see if it's just slightly out of range
movie_rating_index = reindexed_group_ratings.loc[reindexed_group_ratings['movie id'] == movie_id].index[0]
if movie_num < movie_rating_index <= 2*movie_num:
    reasons.append(f"This movie is ranked at position {movie_rating_index}. It will appear if you ask for more recommendations.")
    return reasons
```

Item A does not exists in the database

Item had the same score as another item.

You asked for few items.

Atomic granularity (2/2)

100 peers, 20 closest peers, numPI=5

```
# Find peer ratings of this movie
peer_ratings = ratings.loc[ratings['user id'].isin(peers['user id'])]
movie_ratings = peer_ratings.loc[peer_ratings['item id'] == movie_id]
if movie_ratings.shape[0] == 0:
    reasons.append("This movie is not rated")
    return reasons

if movie_ratings.shape[0] > 0:
    # Find the ratings of the closest peers
    most_similar_peers_rated = movie_ratings.loc[movie_ratings['user id'].isin(closest_peers["user id"])]
    if most_similar_peers_rated.shape[0] < min_rated_num:
        reasons.append(f"Not enough similar peers rated this movie, only {most_similar_peers_rated.shape[0]} out of {min_rated_num} did")
    if movie_ratings.shape[0] < min_rated_num:
        reasons.append(f"Not enough peers rated this movie, only {movie_ratings.shape[0]} out of {min_rated_num} did")
if len(reasons) > 0:
    return reasons
```

No peer has rated this item.

Only x (<numPI) closest peers have rated this

Only x (<numPI) peers have rated this

Group granularity

100 peers, 20 closest peers, numPI=5

```
genre_reasons = []

if all_genres.loc[all_genres['genre'] == genre].shape[0] == 0:
    genre_reasons.append("The genre is not in the database")
    return genre_reasons

# Loop through all movies in the genre and find the reason
# as atomic granularity
reasons = []
for movie_name in movie_names:
    atomic_reasons = atomic_granularity_reason(movies, user_ids, movie_name, ratings, group_ratings, peers, closest_peers, min_rated_num, mov
ie_num)
    reasons = reasons + atomic_reasons

# If a lot of movies have the same reason, that's the general reason for the whole genre
tie_reason = sum(1 if reason.startswith("This movie's predicted rating ties") else 0 for reason in reasons)
low_rank_reason = sum(1 if reason.startswith("This movie is ranked at position") else 0 for reason in reasons)
low_member_score_reason = sum(1 if reason.startswith("A member of the group rated this") else 0 for reason in reasons)
not_rated_reason = sum(1 if reason.startswith("This movie is not rated") else 0 for reason in reasons)
not_enough_similar_peers_reason = sum(1 if reason.startswith("Not enough similar peers rated this movie") else 0 for reason in reasons)
not_enough_peers_reason = sum(1 if reason.startswith("Not enough peers rated this movie") else 0 for reason in reasons)
movie_shoud_in_list = sum(1 if reason.startswith("This movie should be in your recommended list") else 0 for reason in reasons)

main_reason = np.argmax([tie_reason, low_rank_reason, low_member_score_reason, not_rated_reason, not_enough_similar_peers_reason, not_enough_
peers_reason, movie_shoud_in_list])

main_reasons = {
    "0": "Too many tie ratings for movies in this genre",
    "1": f"Movies in this genre are in top {movie_num*2}, show more to see the movies",
    "2": "Your group members rate movies in this genre low score.",
    "3": "Movies of this genre aren't rated much",
    "4": f"Your similar peers do not like {genre}",
    "5": f"Not enough peers rate {genre}",
    "6": "This genre should be in your recommended list"
}

return main_reasons[str(main_reason)]
```

Genre A does not exists in the database

Loop through all movies with the genre and determine the majority reason

Atomic position absenteeism (1/2)

100 peers, 20 closest peers, numPI=10

```
reasons = []

if all_movies.loc[all_movies['movie title'] == movie_name].shape[0] == 0:
    reasons.append("The movie is not in the database")
    return reasons

movie_id = all_movies.loc[all_movies['movie title'] == movie_name]['movie id'].iloc[0]
movie_rating = group_ratings.loc[group_ratings['movie id'] == movie_id]['average'].iloc[0]
same_rated_movies = group_ratings[group_ratings['average'] == movie_rating]

# Find the highest ranking among the similarly rated movies
# to determine if it ties with the movie in question
first_same_rated_movie_id = same_rated_movies.iloc[0]['movie id']
reindexed_group_ratings = group_ratings.reset_index(drop=True)

same_rated_movie_rating_index = reindexed_group_ratings.loc[reindexed_group_ratings['movie id'] == first_same_rated_movie_id].index[0]
if same_rated_movies.shape[0] > 1 and same_rated_movie_rating_index <= movie_num:
    reasons.append("This movie's predicted rating ties with many others")
    return reasons
```

Item A does not exists in the database

Item had the same score as another item.

Atomic position absenteeism (2/2)

100 peers, 20 closest peers, numPI=10

```
# Find the rating of the movie in the desired position (desired rating)
desired_position_movie_rating = reindexed_group_ratings['average'].iloc[desired_position-1]

# Find peer ratings that give a lower score than the desired rating
peer_ratings = ratings.loc[ratings['user id'].isin(peers['user id'])]
movie_ratings = peer_ratings.loc[peer_ratings['item id'] == movie_id]
movie_ratings_low_score = movie_ratings[movie_ratings['rating'] <= desired_position_movie_rating]

# Find closest peers ratings that give a lower score than the desired rating
most_similar_peers_rated = movie_ratings.loc[movie_ratings['user id'].isin(closest_peers["user id"])]
most_similar_peers_rated_low_score = most_similar_peers_rated[most_similar_peers_rated['rating'] <= desired_position_movie_rating]

# Check if more than the threshold number of peers/closest peers give this movie a lower score
# than the desired rating
if most_similar_peers_rated_low_score.shape[0] >= min_bad_rated_similar_peers:
    reasons.append(f"{most_similar_peers_rated_low_score.shape[0]} of {closest_peers.shape[0]}
 most similar peers give this movie a lower score")
if movie_ratings_low_score.shape[0] >= min_bad_rated_similar_peers:
    reasons.append(f"{movie_ratings_low_score.shape[0]} of {peers.shape[0]} peers give this movie a lower score")
if len(reasons) > 0:
    return reasons
else:
    return [f"{peers.shape[0] - movie_ratings_low_score.shape[0]} of peers really like it, but {movie_ratings_low_score.shape[0]}
 do not enjoy it as much"]
```

x of your peers have given a low score to this item.

x of your most similar peers have given a low score to A

x peers like A, but y dislike it.

Asking questions

	movie id	movie title	user2 rating	user17 rating	user35 rating	average
49	50	Star Wars (1977)	5.000000	4.060897	3.267540	4.530449
99	100	Fargo (1996)	5.000000	4.000000	3.855796	4.500000
11	12	Usual Suspects, The (1995)	4.542483	4.126286	2.994006	4.334384
63	64	Shawshank Redemption, The (1994)	4.550331	3.874734	3.221284	4.212532
97	98	Silence of the Lambs, The (1991)	4.407805	3.745008	3.839236	4.076406
22	23	Taxi Driver (1976)	4.292006	3.763017	2.218471	4.027512
44	45	Eat Drink Man Woman (1994)	4.264728	3.762569	3.406358	4.013649
0	1	Toy Story (1995)	4.000000	4.000000	2.549545	4.000000
59	60	Three Colors: Blue (1993)	4.252207	3.735299	2.909408	3.993753
6	7	Twelve Monkeys (1995)	3.973975	4.000000	3.903087	3.986987
55	56	Pulp Fiction (1994)	4.206394	3.710830	3.859936	3.958612
58	59	Three Colors: Red (1994)	4.298380	3.616093	3.019283	3.957236
88	89	Blade Runner (1982)	4.269149	3.515706	4.203631	3.892428
21	22	Braveheart (1995)	4.250651	3.516584	2.386253	3.883617
13	14	Postino, Il (1994)	4.000000	3.741435	3.781842	3.870717
47	48	Hoop Dreams (1994)	4.190924	3.529751	3.313369	3.860338
82	83	Much Ado About Nothing (1993)	4.177120	3.466605	3.343885	3.821863
60	61	Three Colors: White (1994)	4.007329	3.629012	2.512506	3.818171
92	93	Welcome to the Dollhouse (1995)	4.125093	3.438677	2.869367	3.781885
78	79	Fugitive, The (1993)	4.179365	3.308336	2.363283	3.743850

Atomic granularity

- Why not *Mighty Aphrodite (1995)*?

```
atomic_granularity_question("Mighty Aphrodite (1995)", [2, 17, 35], group_ratings)
```

✓ 0.1s

```
Not enough similar peers rated this movie, only 0 out of 5 did
```

Group granularity

```
group_granularity_question("Adventure", [2, 17, 35], group_ratings)
```

✓ 0.1s

```
100it [00:00, 24840.41it/s]

Your similar peers do not like Adventure
```

Atomic position absenteeism

- Why not rank *Toy Story (1995)* second?

```
position_absenteeism_question("Toy Story (1995)", [2, 17, 35], group_ratings, 3)
```

✓ 0.9s

```
37 of 100 peers give this movie a lower score
```

Thanks