Recommendations

Frank Walsh

Problem Statement

- Calculate a list of movie recommendations for a user based on the users ratings.
  - Input: user.id
  - Output: list of movie recommendations
- Use the users movie ratings to find movies.
- General idea:
  - find similar user(s) that like and dislike same movies
  - Recommend unrated movies that the similar users have rated highly

An Approach

User Similarity

- Calculate a **similarity score** for all other users based on ratings
- Store in a List

Find Similar user

- Get the similar User associated with top similarity score
- Get all movies rated by the similar user user

Remove common films

- Remove movies that similar user and current user have both rated

Return recommendations

- Return remaining movies, sorted by movie rating descending.

Calculate Similarity Score

```
private int getSimilarityScore(User user, User
other) {


1.  Get all ratings for user
2.  Get all ratings for other
3.  Get list of movies rated by both users(user and
    other)
4.  Set score = 0
5.  For each movie
    1.  Calculate product of user rating and other
        rating (user.rating*other.rating)
    2.  Add to score
6.  Return Rating


}
```

Guava Functions

- Function<F,T>
  - One way transform from F to T
  - T apply(F input)
- VERY GOOD FOR TRANSFORMING COLLECTIONS

Getting a list of Rating IDs

```
Function<Rating, Long> transform = new Function<Rating, Long>() {
        @Override
        public Long apply(Rating from) {
            return from.id;
        }
    };

    List<Rating> currentMovies = getUserRatings(user.movieId);
    List<Rating> otherMovies = getUserRatings(other.movieId);


// list of movie ids of current users ratings
List<Long> currentMovieIDs = Lists.transform(currentMovies,transform);

// list of movie ids rated by other user
List<Long> otherMovieIDs = Lists.transform(otherMovies,transform);

// common list of rated movies
currentMovieIDs.retainAll(otherMovieIDs);
```