Group Recommender System for Movies

Abhay Joseph[1822329], Ahmed Elzamarany[1823917], Andreea Coaja[1818429], Mariam Marzouk[1754976], and Reng Chiz Der[1826013]

University of Mannheim, 68131 Mannheim, Germany

Abstract. Recommender systems encompass a class of techniques and algorithms that suggest relevant items to users. They predict future behaviors based on past data through a multitude of techniques. The maturity in single-user recommender systems is yet to transfer to group recommender systems. A group recommender system uses the preferences of several users in a group to generate recommendations collectively. Previous survey [1] suggests that current research lacks investigation on the formation of groups, which correspondingly is due to the scarcity of group data across different domains. In this paper, we attempt to tackle two prominent scenarios in group recommendations, grouping of users and performance of large-group recommender system.

Keywords: Content-Based Filtering · Collaborative Filtering · BERT

1 Introduction

Recommender systems collect users' past preferences and create suitable proposal of things to satisfy the users. Traditional recommender systems have concentrated mainly in single-user models. In real life, however, there are numerous circumstances that are generally associated with groups [1]. Factors influencing group recommender systems include social background, social relations, similarities in interests. These can potentially directly or indirectly be extracted from the social network structure [2]. For this paper, we use the dataset from Movielens [3] to form groups based on two distinct criteria and scenarios, resulting in two different group sizes, 3 and 10. Scenario 1 addresses the recommendation for a group of users (3) with commonly watched movies. Scenario 2 tackles the cold-start problem in group recommendation for a larger group of users (10).

An important breakthrough in Natural Language Processing (NLP) is the use of heavily pre-trained transformers for language modelling, such as BERT [4]. These pre-trained Language Models (LMs) are effective for many downstream tasks in NLP such as recommender systems, thus an essential part of this paper. On the whole, we design and implement a hybrid recommender system using Collaborative Filtering with Ranking Factorization Recommender and Content-Based Filtering based on BERT-Embeddings.

The paper is organized into 5 sections. Section 2 overviews data analysis. Section 3 elaborates on the methodology, splitting it into five parts namely-Train-Test Split, Content-Based Filtering, Collaborative Filtering, Aggregation

Strategies, and Hybrid Model. We evaluate our performances in section 4, followed by the conclusion.

2 Data Analysis

2.1 Datasets

2.1.1 MovieLens - The data set, MovieLens 1M Dataset from [3] is used. In particular, we consider the dataset of 1 million anonymous movie ratings of approximately 3,900 movies, by 6040 Movielens users who joined MovieLens in 2000. More recent versions of MovieLens are not preferred as they do not provide user demographic data, critical for our proposed Collaborative Filtering method.

2.1.2 IMDb - Utilizing the IMDb website and web-scraping tool, Beautiful Soup [5], we scrape the significant movie data for each movie in the MovieLens 1M dataset, including genres, director(s), writer(s), year-of-production and cast to complement the dataset. We map these values to the movies in MovieLens based on the IMDb IDs.

2.2 User Groups

For our two scenarios, we create separate best-suited user groups. The 2 approaches are as follows:

2.2.1 Number of Movies Rated (Scenario 1) - We perform KModes Clustering identically to the method used in Scenario 2 to group the users based on number of movies rated by each. Then we filter the groups to make sure there is at least 100 common movies in each group. We result in 102 groups of 3 or 4 users, where each group has at least 100 common movies to perform as test set.

2.2.2 User Demographics (Scenario 2) - Firstly, to find optimal K for clustering, we perform Elbow Method. Subjectively, we observe that 20 is the ideal cutoff point, as shown in Figure 1.

Then, we perform KModes Clustering. KModes Clustering is the equivalent counterpart of KMeans Clustering, best suited for categorical features, to which most user demographic features belong. Besides, KModes is non-parametric, i.e., no distributional assumptions about the data. It circumvents the need to define ad-hoc distance measures on the categorical features in question [6]. Initially, we perform clustering using demographic features, Gender, Age, Occupation and States. Manual inspection of the resulting clusters display unidentifiable clustering patterns. Notably, most clusters consist of users from different states. After dropping States feature, sequential distribution of users is performed on the 20 clusters resulting in a random split, creating groups of 10 or 11 similar users.

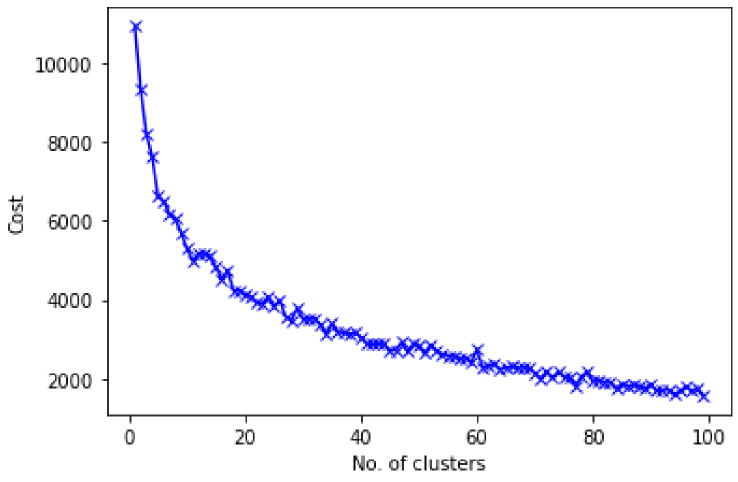


Fig. 1. Elbow Method for Optimal K

2.3 Data Preprocessing and EDA

For the resulting State feature in user demographic data, we utilize the Uszipcode library to convert zip codes into corresponding states. We observe mismatching zip codes, potentially due to typos or foreign users.

We observe no null values or duplicates in all data for MovieLens. Then, preprocessing and analysis are done in several steps. Firstly, data from IMDb is incorporated into the movie database. We also remove movies that are less frequently rated. By removing movies with 5 or fewer total ratings, we have a retention rate of around 87 %. The decision is supported by the smoothing of overall movie ratings' distribution, Figure 2.

From Figure 2, we also observe strong outliers in the right spectrum of the distribution, indicating popular movies. We are particularly interested to investigate the characteristics of frequently rated movies and to determine if these movies would be recommended more frequently by our recommender system.

We suspect that Genre is a significant feature to be considered when applying BERT-Embeddings for Content-based Filtering. Figure 3 demonstrates the distribution of 18 Genres across the dataset. We observe that Drama, Comedy, and Romance has highest frequencies. Investigating demographic features lead us to believe that popular genres are associated highly with the age group and occupations of the user. Overall, we observe that all genres, except Horror, possess left-skewed distribution (with a mean of 3.5), indicating the correlation between liking and ratings.

User demographic features are incorporated as side features for the Collaborative Filtering algorithm. From Figure **??** observe that the most frequent age group is 25-34, accounting for approximately 2000 users in our dataset. The remaining users are distributed equally among other age groups, for teenagers to adults. Correspondingly, the most frequent occupation groups include college students and executive/managers.

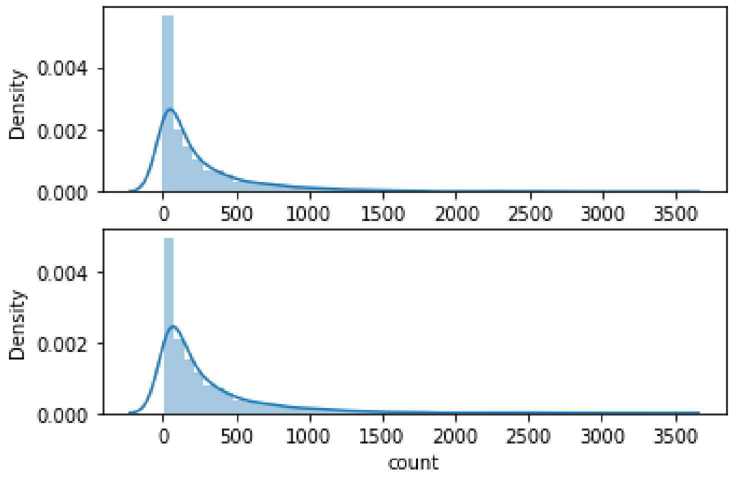
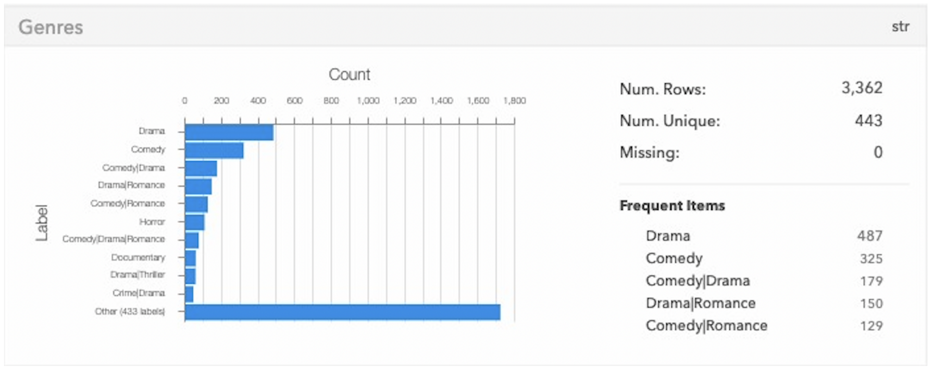


Fig. 2. Rating Distribution

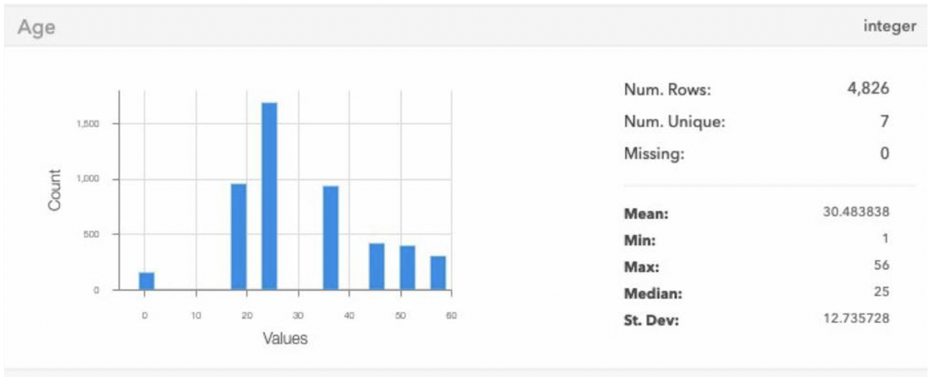


Num. Rows:	3,362
Num. Unique:	443
Missing:	0

Frequent Items

Drama	487
Comedy	325
Comedy|Drama	179
Drama|Romance	150
Comedy|Romance	129

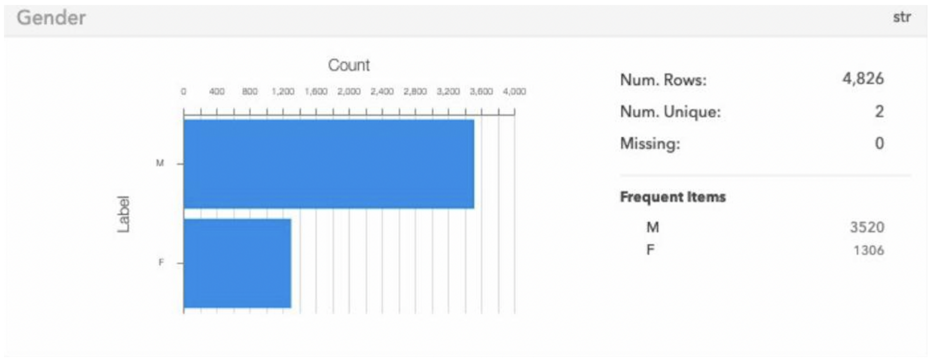
Fig. 3. Movie genres.



Age integer

Num. Rows:	4,826
Num. Unique:	7
Missing:	0
Mean:	30.483838
Min:	1
Max:	56
Median:	25
St. Dev:	12.735728

Fig. 4. Age distribution.



Gender str

Num. Rows:	4,826
Num. Unique:	2
Missing:	0

Frequent Items

M	3520
F	1306

Fig. 5. Gender distribution.

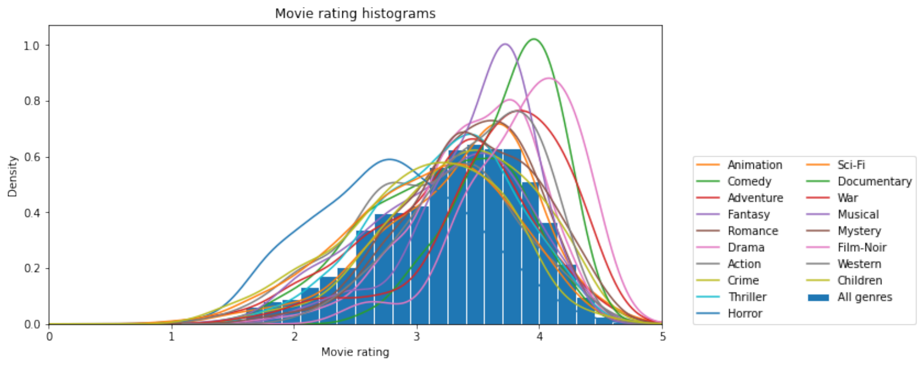


Fig. 6. Movie's ratings by genre.

3 Methodology

3.1 Scenario 1 - Improving Group Recommendation Accuracy

3.1.1 Approach Description - The scarcity of data across domains for group recommender systems has been plaguing the development of such systems. In our case of offline evaluation, i.e., defining user's preferences through past behaviors [7], we have limited data to evaluate the performance of our models. On the other hand, online evaluation does not suffer from a lack of data by testing real-time scenarios. To overcome this dilemma, we create smaller groups of users(3) to ensure the existence of overlapping movie ratings.

3.1.2 Train-Test Split - Instead of splitting the data traditionally, test data for each group consists of ratings for the commonly rated movies by members of each group. The remaining movie ratings by each user are the training set.

3.1.3 Comparison with Baselines - For Scenario 1, group recommender systems using single model, i.e., either Content-based Filtering or Collaborative Filtering act as the baselines for the aggregation strategies and hybrid model.

3.2 Scenario 2 - Tackling Cold-Start Problem for Large Groups

3.2.1 Approach Description - Collaborative Filtering recommenders suffer from cold-start problems, i.e., low prediction accuracy when new users or items enter the system. Consequently, group recommender systems utilizing Collaborative Filtering algorithms inherit such problem [8]. Therefore, we aim to tackle such problem with the combination of two approaches. We incorporate side features for users such as demographic features in our Collaborative Filtering approach. The result is then combined with result from Content-based Filtering using BERT-Embeddings forming the basis of our hybrid model.

3.2.2 Train-Test Split - We do standard train-test split of 80:20 among users, ensuring that members from the same group belong to either training or testing data.

3.2.3 Comparison with Baselines - For Scenario 2, single-user recommender systems, i.e. Content-based Filtering and Collaborative Filtering act as the baselines for the group recommender system. We aim to achieve similar accuracy in group recommender systems, indicating the aggregation strategies and hybrid model are successful.

3.3 Content-based Filtering

For the content-based approach, we represent each movie by its corresponding metadata, which comprises of these features:

Movie Features

Feature	MovieLens	IMDb
Title:	✓	✓
Genres:	✓	✓
Director:	☐	✓
Writer:	☐	✓
Stars:	☐	✓
Description:	✓	✓

*Description is used only in Scenario 1

BERT makes use of Transformer, an attention mechanism that learn contextual relations between words (or sub-words) in a sentence. For our use-case, we apply BERT-Embeddings to both the movie and user data. All movies' metadata are transformed using BERT-Embeddings to have representation for each. On the other hand, a user's watch histories are merged and transformed into encodings to create a user profile. Finally, we apply cosine similarity between user profile and all movie representations to output a list of most similar movies to the user, thus predicting the best recommendation for each user.

3.4 Collaborative Filtering

Collaborative filtering (CF) is a method of making automatic predictions about the interest of a user by collecting preferences or taste information from other users (collaborating the interest) In order to establish recommendations, CF systems need to relate two fundamentally different entities: items and users. There are two primary approaches to facilitate such a comparison, which constitute the two main techniques of CF: the neighborhood approach (Memory Based) and latent factor models (Model Based)[9]. Latent factor models, including our model from TuriCreate, transform both items and users to the same latent factor. The latent space tries to explain ratings by characterizing both item and users on factors automatically inferred from user feedback (also known as item- and user-embeddings)[10].

We utilize the TuriCreate library by Apple Inc. for the Collaborative Filtering approach. We first allow TuriCreate to determine the optimal model and evaluation metric (RMSE vs Precision-Recall) based on our input data and target variable. Our final Collaborative Filtering comprises of a Ranking Factorization recommender. The recommender learns latent factors for each user and item and uses them to rank recommended items according to likelihood of observation. Such model suits our use-case of having explicit ratings which both rating and ranking prediction are desired for upstream usage of hybrid model.

3.4.1 Scenario 2 - For Scenario 2, in order to identify the optimum model, we split the dataset (training data) into training and validation set. We conduct a random split by each user's movie ratings. To further improve the result of ranks for our model, we further pre-process the split to ensure that only high ratings (≥ 4) are included in the validation set.

Four models are generated using different criteria. **Model Item-Similarity**, which performs as the baseline is an item similarity model. It naively ranks an item according to its similarity to other items observed by an user. **Model Basic** perform a Ranking Factorization recommender without user features which **Model User-Feature** utilizes. User features are incorporated as shown in Figure . Three user side features that we suspect are significant to profile a user are Gender, Age and Occupation. Zip-Code is again omitted here for having noisy data. Lastly, we attempt to finetune the model by utilizing additional hyperparameters including regularization on ranking and setting unobserved rating value to be 1, forming **Model Finetuned**.

The predicted score for user i on item j is given by:

$$score(i,j) = \mu + w_i + w_j + a^T x_i + b^T y_i + u_i^T v_j$$

where μ is a global bias term, w_i is the weight term for user i, w_j is the weight term for item j, x_i and y_i are respectively the user and item side feature vectors, and a and b are respectively the weight vectors for those side features. The latent factors are given by u_i and v_j.

3.5 Aggregation

The main task in a group recommender system is to design a model that integrates all the preferences of the group members to predict the ratings for a group of users. This integration is done with the help of different aggregation strategies to combine the different ratings of the individual users (and groups for Scenario 2) to get one unified rating for the whole group and is chosen according to the requirements of the group.

Three main aggregation methods were used:

- Average: simply taking the average of the individual users.
- Least misery: taking the least rating to represent the rating for the whole group
- Most pleasure: taking the highest rating to represent the rating for the whole group
- Average without misery: taking the average after excluding the lowest rating among the group (change: w/o misery means below threshold)

These aggregation methods were used on the group level and with the hybrid model.

3.6 Hybrid

In order to improve recommendation accuracy and user's satisfaction, and to solve the problems of scalability, cold start, and data sparse, many traditional technologies are combined with each other. For the Hybrid approach, the ratings generated between Collaborative Filtering and Content-based Filtering are simply combined using one of the aggregation methods mentioned in the subsection 3.5. One of the reasons is Average strategy performs relatively well compared to other Hybrid Recommender strategy.

4 Evaluation

4.1 Scenario 1 - Groups Evaluation

For each model we evaluated based on two metrics: accuracy and F1 score. The evaluation is done particularly on the common movies between each group. Same aggregation steps applied on the rating predicted for the users in the group are also applied on the gold standard ratings. Before calculating the metrics, we convert the ratings into binary target variable by assigning 1 if the rating is high ((≥ 4) and 0 otherwise.

Table 3 displays the results after applying different aggregation strategies to the outputs of the models. The outputs of the content-based and the collaborative filtering methods are similar, except for Most Pleasure Aggregation Strategy. Least Misery produces highest accuracy but lowest F1 score, which is not ideal, taking false negatives and false positives into consideration. The highest F1 score is achieved by Most Pleasure Aggregation, but also with the lowest accuracy. Using Average or Average Without Misery results in more balanced results. The hybrid model generates the best combined results.

However, even the best model yields weak results. We expect that it is due to the small dataset available. Merely 102 groups with 100 common movie were available for testing.

It tends to predict ratings with less accuracy than the non-ranking factorization-recommender, but it tends to do much better at choosing items that a user would rate highly. This is because it also penalizes the predicted rating of items that are significantly different from the items a user has interacted with. In other words, it only predicts a high rating for user-item pairs in which it predicts a high rating and is confident in that prediction [6].

4.2 Scenario 2 - Large Group Evaluation

For the Single-User Collaborative Filtering, we observe that by incorporating both user demographic data and finetuning, we achieve the best Root Mean Squared Error(RMSE) for test data, according to Table 2. However, further investigation of relevance of each additional user feature is needed to determine their correlations to user profiling and movie ratings.

Table 1. Group recommendation results.

Aggregation method	Content-based		Collaborative filtering		Hybrid	
	Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score
Average	57.48%	2.7%	57.98%	3.24%	57.98%	3.24%
Least misery	73.58%	0.7%	73.68%	0.27%	73.68%	0.27%
Most pleasure	19.66%	16.82%	30.46%	35.14%	30.46%	35.13%
Average without misery	32.85%	6.49%	35.57%	13.60%	35.57%	13.60%

We perform testing on both single-user recommender systems and on group recommender systems with different aggregation strategies. From the RMSE for test data, we observe that Least Misery performs worst. Average and Average-Without-Misery produce similar performance due to the threshold set for Average-Without-Misery. We conclude that in the case of movie ratings prediction, Average-Without-Misery may not be too suited for the use case. Most Pleasure Aggregation Strategy results in lowest RMSE. Overall, our hybrid models with different aggregation strategies perform comparatively well to single-user recommender system. This may be credited to successful clustering, and efficiency of the respective models. However, we observe that the predicted ratings have interquartile range in [2, 4] that may have resulted in positive RMSE performance.

Table 2. Scenario 2- Performance of Single-User Collaborative Filtering

Models	**Train RMSE**	**Test RMSE**
Basic	0.78	1.19
User-Features	1.03	1.21
Item-Similarity	N/A	4.39
Finetuned	1.05	1.07

Table 3. Comparison of Accuracy for Single-User and Group Algorithms.

Users	**Method**	**RMSE**
Single-User	Content-based	1.0992
Single-User	Collaborative Filtering	1.0827
Group	Average Without Misery	1.0677
Group	Most Pleasure	1.0567
Group	Least Misery	1.1231
Group	Average	1.06682

5 Conclusion

Two main motivation are tested, creating recommendation on users groups and creating models to predict movies for new users. We use different grouping methods for each. For Scenario 1, we group by the number of movies rated to result to 102 groups of 3 users with at least 100 common movies for testing. For Scenario 2, we group the users based on their demographics into groups of 10, which resulted in approximately 500 group. Hybrid Group Recommender performs better, but with rooms for improvement.For future work, different aggregation strategies and hybrid recommender systems could be investigated. New data collection targeted for group recommendation would also be beneficial to the development of group recommender systems.

References

1. Sriharsha Dara, C. Ravindranath Chowdary, and Ch. Naveen Kumar. A survey on group recommender systems. *Journal of Intelligent Information Systems*, 54:271–295, 2019.
2. Michal Kompan and Mária Bieliková. Group recommendations: Survey and perspectives. *Comput. Informatics*, 33:446–476, 2014.
3. F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context, 2015.
4. Gustavo Penha and Claudia Hauff. What does bert know about books, movies and music? probing bert for conversational recommendation. *CoRR*, abs/2007.15356, 2020.
5. Leonard Richardson. Beautiful soup documentation. *April*, 2007.
6. Anil Chaturvedi, Paul E. Green, and J. Douglas Caroll. K-modes clustering. *J. Classif.*, 18(1):35–55, jan 2001.
7. Ladislav Peska and Peter Vojtás. Off-line vs. on-line evaluation of recommender systems in small e-commerce. *CoRR*, abs/1809.03186, 2018.
8. Lara Quijano-Sanchez, Derek Bridge, Belén Díaz-Agudo, and Juan Recio-García. A case-based solution to the cold-start problem in group recommenders. pages 3042–3046, 08 2013.
9. Yehuda Koren and Robert Bell. *Advances in Collaborative Filtering*, pages 77–118. 01 2015.
10. Apple Inc. Turi create api, 2020.