HYBRID MOVIE RECOMMENDER SYSTEM

Badai Kesuma badai@kth.se

Hamid Dimyati dimyati@kth.se

Jie Liu ljie@kth.se

Mostafa Abdelrahman mosabd@kth.se

May 24, 2020

ABSTRACT

Recommender systems provide suggestions of data items likely to be of interest to the user by exploiting different techniques. In this project, we attempt to systematically evaluate different recommendation techniques for movies using MovieLens 100K dataset. We investigated a total of six different techniques; 1) content-based filtering, 2) item-item collaborative filtering, 3) matrix factorization collaborative filtering, 4) weighted hybrid filtering, 5) mixed hybrid filtering, 6) cascade hybrid filtering. The investigation finds that matrix factorization collaborative filtering achieves higher scores for NDCG than content-based filtering. In addition, weighted hybrid recommendation techniques could reduce much RMSE with the cost of a little NDGC reduction.

Keywords recommender system · collaborative filtering · content-based filtering · hybrid approach

1 Introduction

A recommender system has been developed long before the internet became widely used by people. One of the earliest recommender systems was implemented for Usenet, a worldwide distributed discussion system or commonly known as a predecessor of the World Wide Web. A collaborative filtering system, called GroupLens, was developed to make recommendations of netnews to the Usenet users by Resnick et al. [1]. It was inspired by Tapestry, a filtering system of a large volume of email, where the term collaborative filtering was introduced [2]. The movie recommendation system is also one of the most favorite cases for research development. There are several alternative datasets for building a movie recommender system such as EachMovie, MovieLens, or Netflix. One of the challenges in building recommender systems with these movie datasets is the low density of the user rating matrix. Specifically, the MovieLens only has a 1.70% density level, which makes collaborative filtering difficult to reveal the most optimum results. Moreover, another weakness of the collaborative filtering algorithm is the inability to recommend new movies or new users, commonly called as cold-start problems. Balabanović and Shoham [3] also mentioned that collaborative filtering algorithms could also potentially produce poor recommendations due to users that have unusual tastes compared to the rest of the population.

Another way to build a movie recommender system is to use the availability of movie information such as genre, release year, directors, and actors, which is commonly called as content-based filtering. However, those movie attributes cannot represent the user preferences entirely and, as stated by Balabanović and Shoham [3], this method is high likely to suffer from over-specialization where the system restricts the user to see only items that have similar contents to those already rated. A hybrid approach by combining the collaborative filtering and content-based filtering comes as a solution to build better movie recommender systems. In this paper, we would present how much improvement the hybrid approach could produce in recommending a movie compared to the basic collaborative filtering and content-based filtering.

2 Related Work

Hybrid recommendation systems were introduced by Balabanović and Shoham [3], where they compare user profiles to determine similarity, and the users are recommended items that are highly rated by similar profiles. There are all seven possible approaches of hybridization of all five common recommendation techniques that are collaborative filtering, content-based, demographic-based, and knowledge-based or utility-based filtering, in which the hybridization approaches comprise: weighted, switching, mixed, feature combination, cascade, feature augmentation, and meta-level [4].

1. **Weighted:** The scores (or votes) of several recommendation techniques are combined to produce a single recommendation, for example, a linear combination of recommendation scores.
2. **Switching:** The system switches between recommendation techniques depending on the current situation. For example, Burke [4] mentioned that the DailyLearner system applies a content or collaborative hybrid where a content-based recommendation technique is used first. If the content-based system cannot be able to make a recommendation, then the collaborative recommendation will be involved.
3. **Mixed:** Recommendations from several different recommenders are presented at the same time. For example, Burke [4] introduced the PTV system that applies a mixed approach for recommending programs from TV comments. It employs a content-based approach on the descriptions of TV shows and also a collaborative information approach for user preferences. The final recommended programs combine both techniques together. Mixed hybrid recommender systems are simple and can eliminate the cold-start (new user or new item) problem.
4. **Feature combination:** Features from different recommendation data sources are thrown together into a single recommendation algorithm. For example, the embeddings of the collaborative filter recommender can be added as additional features to a content-based recommender.
5. **Cascade:** Burke [4] introduced the technique to create a rough result from one recommendation technique and fine-tune the result through a second recommendation technique. In this technique, the system is prevented from recommending low-rated items.
6. **Feature augmentation:** The rating output from one technique is used as an input feature to another. This approach is different from the cascade approach because, in the cascade approach, recommendations are combined together while in this approach, the output of one system is used as an input to another.
7. **Meta-level:** The model learned by one recommender is used as input to another. This approach is different from feature augmentation because the entire model is used as input, not just the model output.

Using the Entree dataset of the restaurant, Burke [5] showed an empirical study of notable and stable performance of feature augmentation and cascade method by combining content-based or knowledge-based to collaborative filtering (order sensitive) over other possible methods. Melville et al. [6] introduces an algorithm called CBCF (Content-Boosted Collaborative Filtering) to EachMovie dataset by creating a pseudo-user-ratings matrix that contains the users' actual ratings and content-based predictions for the unrated items. This matrix aims to deal with the common problem in a collaborative filtering algorithm that suffers from a sparse matrix of user-ratings. In the end, the predictions are made based on this dense pseudo-user-ratings matrix.

Basu et al. [7] took a different approach to build hybrid recommender systems by modeling the problem as a machine learning (classification) problem. The main algorithm in their hybrid recommendation system, using a combination of collaborative and content features, tried to classify a set of users, whether to like or dislike a given set of movies. Basilico and Hofmann [8] continued developing this kind of approach by inventing JRank (joint kernel perceptron learning), which adapts the concept of ordinal regression by assuming the rating as a target variable and incorporating multiple kernels as the features.

Our proposed system uses a hybrid approach. Firstly, the ratings of unrated items are predicted based on the available information using some recommendation algorithms. Secondly, the system finds items that maximize the user's utility based on the predicted ratings and recommends them to the user. We construct a movie graph for the similarity of movies based on their characteristics such as release year, genre, runtime, directors, and writers which are extracted from the IMDb dataset.

3 Methodology

3.1 Data Description

The datasets used for this paper are from MovieLens [9] and IMDb [1]. The MovieLens dataset contains 100,836 ratings of 9,742 movies from 610 users created between March 29, 1996 and September 24, 2018. The ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars) from randomly selected users that have rated at least 20 movies.

[1]https://datasets.imdbws.com/

Table 1: Summary of datasets

Dataset	Features	Summary
MovieLens	Rating density	1.70%
	Avg. clustering coefficient	0.17
	Type of distribution	Power-law
IMDb	Type	9 unique values
	Genre	19 unique values
	Release year	1902-2018
	Runtime (min)	2-680 average: 105
	Directors	4,290 unique values
	Writers	8,274 unique values

The IMDb dataset contains information about at least 9,916,856 movies and is updated daily. For this dataset, we extracted features such as title, genre, type, release year, runtimes, directors, and writers of the movies based on selected movies in the MovieLens dataset using imdbId as a foreign key. A '\N' is used to denote that a particular field is missing or null for that attributes. The summary of these two datasets as used in our experiments are summarized in Table 1.

3.2 Data Preprocessing

Before recommendations can be made, the movie data is processed into separate profiles, one for each movie's rating given by the user, defining that user's movie preferences. As an initial step, the MovieLens dataset is combined with IMDb dataset to get more descriptions of the movie that the users have rated. This combined dataset is then set as features, which passes several processes, including data preparations and feature selections, to be used for content-based filtering and collaborative filtering.

Data cleaning is part of a preprocessing step, which is extremely important. It can be applied to remove missing values. In the movie description for content-based filtering, missing data can result in loss of efficiency, less information extracted from data or conclusions statistically less strong, a complication in handling and analyzing the data where methods are in general not prepared to handle them and may have an impact on modeling, and sometimes they can destroy it. Furthermore, missing data can introduce bias resulting from differences between missing and complete data.

Considering a large number of movies and missing values in the dataset, in this experiment, we implement missing data ignoring techniques, listwise deletion, by simply omitting the cases that contain missing data in the numerical features. In listwise deletion, a case is dropped from the analysis because it has a missing value in at least one of the specified variables. The analysis is only run on cases that have a complete set of data.

In the recommender system, categorical features are common and often of high cardinality. In this experiment, we use one-hot encoding to transform categorical features in which each category value is converted into a new column and assigned a 1 or 0 (notation for true/false) value to the column. In the case of missing data in categorical data, all new columns of each categorical value are assigned to 0.

3.3 Algorithm Description

In our approach, we combine two recommendation algorithms: Content-based Filtering and Collaborative Filtering. Approaches for combining the algorithms are discussed in section 3.3.3.

3.3.1 Content-based Filtering

In Content-based Filtering, hand-engineered features are used to construct a numerical vector representation of the item to be recommended. These features can be numerical (e.g., movie runtime, release year) represented as in vectors or categorical (e.g., genres, actors), which are one-hot encoded to obtain a numerical representation.

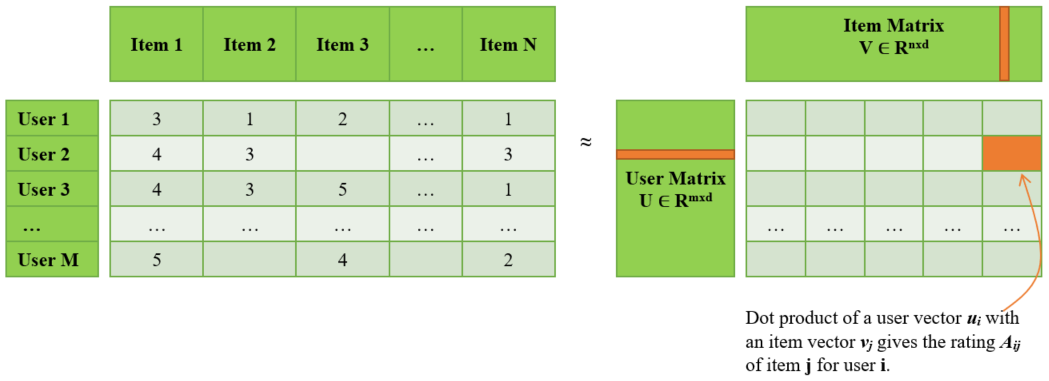

Figure 1: Collaborative Filtering using Matrix Factorization.

Once each item is represented as a vector in the feature space, the similarity between two items is calculated using cosine similarity. The cosine similarity produces a value between -1 (indicating exact opposites) to +1 (indicating exact similarity). It is calculated between two vectors u and v according to equation (1).

$$sim(u, v) = \frac{u.v}{\|u\|\|v\|} \tag{1}$$

In Content-based Filtering, we only consider the recommendations of a single user disregarding all other users. Let the set $N(u)$ be the set of all movies rated by a user u and $\overline{r}_u$ is the average rating for user u. The estimated user's rating for a new movie is calculated according to equation (2).

$$r_i = \overline{r}_u + \frac{\sum_{j \in N(u)} sim(i, j) * (r_j - \overline{r}_u)}{\sum_{j \in N(u)} sim(i, j)} \tag{2}$$

3.3.2 Collaborative Filtering

Collaborative Filtering relies on ratings provided by other users to recommend items for a specific user. The intuition is that if two people have similar ratings for a set of items, then they are more likely to share similar ratings for new items. Collaborative filtering algorithms can be broadly classified into two classes of methods: model based (e.g. matrix factorization) and memory based (e.g. item-item collaborative filtering).

Matrix Factorization Collaborative Filtering One technique for Collaborative Filtering is Matrix Factorization, which tries to simultaneously learn the users and items vector representation in the feature space without relying on hand-engineered features.

Given the rating matrix, $A \in R^{m \times n}$ containing ratings of different items given by different users where m is the number of users, and n is the number of items. The algorithm learns two matrices U and V such that $A = UV^T$ where $U \in R^{m \times d}$ is the user matrix where each row is the vector representation of a user, and $V \in R^{n \times d}$ is the item matrix where each row is the vector representation of an item. The dimension d is a parameter of the algorithm used to specify the number of features (hidden factors) to learn per item.

The Matrix Factorization problem can be reformulated as an optimization problem that minimizes the distance between the rating matrix A and the product UV^T. The optimization function is described in equation (3) where the Frobenius norm is defined as $\|A\|_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$

$$\min_{U \in R^{m \times d}, V \in R^{n \times d}} \|A - UV^T\|_F^2 \tag{3}$$

Additional regularization terms can be added to the loss function:

$$\min_{U \in R^{m \times d}, V \in R^{n \times d}} \|A - UV^T\|_F^2 + \alpha \times \rho(\|U\|_1 + \|V\|_1) + \alpha(1-\rho)(\|U\|_F^2 + \|V\|_F^2) \tag{4}$$

Where ρ parameter control the combination of L1 and L2 regularization and α parameter control the amount of regularization.

Once the vector representations are learned, an unrated movie by a specific user u can be predicted using the dot product between the user vector u_i and the item vector v_i.

Item-Item Collaborative Filtering This technique consist of: (1) calculating the similarity among items (e.g. cosine similarity as defined in equation 1 based on the vector representation of each item and (2) calculate the rated prediction for a new item (e.g. weighted sum as defined in equation 2). At first glance, this seems identical to content-based filtering technique described in section 3.3.1. However, in content-based filtering, the vector representation is driven from the actual content of an individual item disregarding other users' interactions but in item-item collaborative filtering the content of the item are not considered and the vector representations of each item are columns of the rating matrix in order to integrate other users ratings in the evaluation process.

3.3.3 Hybrid approach

In order to combine the ratings calculated by the two methods described above, we describe three alternative solutions that we evaluate in our experiments.

Mixed: The top K ranked items are selected by content-based filtering algorithm and the rest of the recommendations are generated from collaborative filtering.

Weighted Average: The ratings are weighted and combined to produce a single rating score per item, which is used to rank the top K items.

$$r_i = \alpha * r_{i,content-based} + (1-\alpha) * r_{i,collaborative-filtering} \tag{5}$$

where $0 \le \alpha \le 1$

Cascade: At each stage, the recommendations are filtered so that only items with rating above a certain threshold τ are recommended. The recommendation $rec_k(u,i)$ at stage k for user u and item i is defined as:

$$rec_k(u,i) = \begin{cases} rec_k(u,i) & \text{if } rec_{k-1}(u,i) > \tau \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

3.4 Evaluation

We compare the results of F1-score and NDCG (Normalized Discounted Cumulative Gain) across the proposed hybrid approached (mixed, weighted average and cascade) and the baseline methods (content-based filtering and collaborative filtering). The dataset was split into 80% for training the models and 20% for testing and evaluating the performance. In our experiment, we use F1@K and NDCG@K for evaluation, where K is the number of top-K items in the result.

Let R_K be the ground truth of the top-K rated items in the testing dataset where rel_i is the true rating given to item i by the user and R be the top-K rated items returned by the algorithm where r_i be the rating given to item i by the algorithm. Also, let REL_K be the top-K items sorted according to their true rating.

The F1@K is calculated as follows:

$$F1@K = \frac{2 * precision * recall}{precision + recall} \tag{7}$$

where $precision = \frac{|R \cap R_K|}{|R|}$ and $recall = \frac{|R \cap R_K|}{|R_K|}$

Similarly, NDCG@K is calculated as follows:

$$NDCG@K = \frac{DCG}{IDCG} \tag{8}$$

where $DCG = \sum_{i=1}^{K} \frac{r_i}{\log_2(i+1)}$ and $IDCG = \sum_{j \in REL_K} \frac{rel_i}{\log_2(j+1)}$

4 Results

Hyperparameter tuning of each algorithm was performed using a Randomized Grid Search, where parameter configurations are randomly sampled from a distribution. The best parameters were selected based on the NDCG score for each algorithm. Table 2 shows the distribution used to form the Search Grid. In total 10 combinations are sampled.

Table 2: Hyperparameter tuning

Algorithm	Hyperparameter	Description	Distribution
Collaborative filtering (Matrix factorization)	α	Amount of regularization in 4	uniform(0,1)
	ρ	Combination of regularization in 4	uniform(0,1)
	$n\_components$	Number of latent factors to learn	randInt(50,150)
Weighted hybrid filtering	α	Weighting parameter in 5	uniform(0.2,0.8)
Cascade hybrid filtering	τ	Threshold parameter in 6	uniform(0.5,5)
Mixed hybrid filtering	k	Number of items to pick from first recommendation algorithm	uniform(5,50)

The results of hyperparameter tuning indicate that the best NDCG would be obtained at $n\_components$ = 70 for matrix factorization collaborative filtering, similarly, the optimal parameters for the hybrid algorithms were α = 0.7 for the weighted algorithm, τ = 4.54 for the cascade algorithm, and K = 6 for the mixed algorithm.

The dataset was split into 80:20 as detailed in section 3.4 in order to compare the performance of each algorithm (content-based filtering, item-item collaborative filtering, matrix factorization collaborative filtering, weighted hybrid filtering, mixed hybrid filtering, and cascade hybrid filtering) using RMSE, Precision@K, Recall@K, F1@K and NDSG@K with K at 10, 25 and 100. The results of our experiments are summarized in Table 3.

Table 3: Model Performance Comparison

Metrics	Algorithms					
	Content-based	Collaborative (item-item)	Collaborative (MF)	Hybrid (Weighted)	Hybrid (Cascade)	Hybrid (Mixed)
RMSE	**0.91178**	0.91632	3.24764	3.14003	3.60836	3.24545
NDCG@10	0.01019	0.00000	0.52613	**0.52637**	0.02358	0.23599
Precision@10	0.00264	0.00000	0.23795	**0.23952**	0.00503	0.12182
Recall@10	0.00053	0.00000	0.14989	**0.15020**	0.00084	0.08211
F1@10	0.00088	0.00000	0.18392	**0.18462**	0.00143	0.09810
NDCG@25	0.02239	0.00035	0.52655	**0.52847**	0.02988	0.33561
Precision@25	0.00363	0.00007	0.17652	**0.17685**	0.00464	0.14850
Recall@25	0.00206	0.00001	**0.26039**	0.25989	0.00232	0.22176
F1@25	0.00262	0.00002	0.21040	**0.21047**	0.00309	0.17787
NDCG@100	0.05015	0.00100	0.51624	**0.51711**	0.04413	0.38443
Precision@100	0.00432	0.00007	0.09033	**0.09079**	0.00403	0.08809
Recall@100	0.01146	0.00003	**0.45658**	0.45535	0.01018	0.44762
F1@100	0.00628	0.00004	0.15083	**0.15140**	0.00577	0.14721

Content-based filtering and item-item collaborative filtering algorithms had the lowest overall RMSE score among all methods at 0.911 and 0.916 respectively. However, they suffered low NDCG and F1 scores, specifically item-item collaborative filtering had almost zero scores for NDCG. Matrix-factorization collaborative filtering performed better in terms of NDCG and F1 for different values of K (10, 25, 100). However, it performed poorly in RMSE at 3.24.

The weighted hybrid method had the best overall performance among hybrid methods and the mixed hybrid method came a close second. In particular, the weighted method had the highest NDCG scores for all values of K (10, 25, 100) while it had a slightly lower RMSE than matrix factorization.

5 Conclusions

In this experiment, we investigated six recommendation techniques that suggest movies using MovieLens 100K dataset. We tried to evaluate suggestion relevancies, the difference between the actual value and the predicted value, and the accuracy of the prediction to find the best technique. We also optimized parameters for each technique according to training data in our experiments.

Three general results can be deduced from our experiment. First, content-based filtering and collaborative filtering are the opposite in terms of their own strength and weakness. It can be seen from the evaluation result that content-based filtering has the best RMSE score, which is the lowest compared to other techniques, and the worst NDCG score where the value is close to zero. On the other hand, collaborative filtering achieved the best NDCG score while having a very poor RMSE score. Second, our experiments have shown that weighted and mixed hybrid recommendation techniques generally outperform other hybrid recommendation techniques and simultaneously balancing its evaluation scores compared to our base model. In particular, the weighted method managed to achieve a lower RMSE and the best overall performance in NDCG and F1 scores. Third, we can not find a single method that has the best performance in all three evaluation scores. In our opinion, the end-user cares about NDCG more than RMSE because end users will have a pleasant experience if the top recommendations are aligned with their taste. Hence, the actual rating predictions are not as important as the relative order of items.

Limitations and challenges Using one-hot encoding for feature representation have led to very high dimensional vector representations, causing memory and computability concerns for the algorithms. In addition, due to the difficulty of manual data annotation, in our experiment, only movies in the test set (movies watched and rated by a user) were assumed to be relevant, which caused the generally low NDCG.

References

[1] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[2] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[3] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[4] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[5] Robin Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.

[6] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai*, 23:187–192, 2002.

[7] Chumki Basu, Haym Hirsh, William Cohen, et al. Recommendation as classification: Using social and content-based information in recommendation. In *Aaai/iaai*, pages 714–720, 1998.

[8] Justin Basilico and Thomas Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*, page 9, 2004.

[9] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

Individual Contribution

Badai Kesuma (badai@kth.se)	Data Description Data Preprocessing Mixed Hybrid Filtering
Hamid Dimyati (dimyati@kth.se)	Related Work Summary Statistics Cascade Hybrid Filtering Introduction + document merging
Jie Liu (ljie@kth.se)	Related Work Item-Item Collaborative Filtering
Mostafa Abdelrahman (mosabd@kth.se)	Content-Based Filtering Matrix Factorization Collaborative Filtering Hyperparamter tuning