TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 02/12 pp180–191
DOI: 10.26599/TST.2018.9010118
Volume 25, Number 2, April 2020

Personalized Real-Time Movie Recommendation System: Practical Prototype and Evaluation

Jiang Zhang, Yufeng Wang*, Zhiyuan Yuan, and Qun Jin

Abstract: With the eruption of big data, practical recommendation schemes are now very important in various fields, including e-commerce, social networks, and a number of web-based services. Nowadays, there exist many personalized movie recommendation schemes utilizing publicly available movie datasets (e.g., MovieLens and Netflix), and returning improved performance metrics (e.g., Root-Mean-Square Error (RMSE)). However, two fundamental issues faced by movie recommendation systems are still neglected: first, scalability, and second, practical usage feedback and verification based on real implementation. In particular, Collaborative Filtering (CF) is one of the major prevailing techniques for implementing recommendation systems. However, traditional CF schemes suffer from a time complexity problem, which makes them bad candidates for real-world recommendation systems. In this paper, we address these two issues. Firstly, a simple but high-efficient recommendation algorithm is proposed, which exploits users' profile attributes to partition them into several clusters. For each cluster, a virtual opinion leader is conceived to represent the whole cluster, such that the dimension of the original user-item matrix can be significantly reduced, then a Weighted Slope One-VU method is designed and applied to the virtual opinion leader-item matrix to obtain the recommendation results. Compared to traditional clustering-based CF recommendation schemes, our method can significantly reduce the time complexity, while achieving comparable recommendation performance. Furthermore, we have constructed a real personalized web-based movie recommendation system, MovieWatch, opened it to the public, collected user feedback on recommendations, and evaluated the feasibility and accuracy of our system based on this real-world data.

Key words: movie recommendation system; collaborative filtering; real-time; virtual opinion leader; data mining

1 Introduction

With the growth of big data generation across various fields, information overload is becoming a critical problem. To address this, a number of Recommendation Systems (RS) have been developed to help consumers find items of interest and decide between products among huge databases. These systems have been applied to various products and services on the internet, including film and video, music, social networking, reading, news, and personalized e-mail and advertising. Three of the domains in which RSs are used most frequently are movies, documents, and product reviews, mainly because of the ease in accessing test data[1–4]. For instance, in the movie domain, MovieLens and Netflix are two online datasets of movie ratings. The movie industry is a prodigious producer of video. Already by the year 2000, it was reported that more

• Jiang Zhang, Yufeng Wang, and Zhiyuan Yuan are with the Nanjing University of Posts and Telecommunications (NUPT), Nanjing 210003, China. E-mail: 1217012225@njupt.edu.cn; wfwang@njupt.edu.cn; 2361660127@qq.com.
• Qun Jin is with the Networked Information Systems Laboratory, Department of Human Informatics and Cognitive Sciences, Faculty of Human Sciences, Waseda University, Tokyo 163-8001, Japan. E-mail: jipn@waseda.jp.
* To whom correspondence should be addressed.
Manuscript received: 2018-06-03; revised: 2018-08-23; accepted: 2018-09-01

than 4500 movies are released every year around the world, spanning approximately 9000 hours of video. With such a massive amount of choice, there is great demand for technologies that enable viewers to access movies of interest conveniently and therefore facilitate movie distribution.

1.1 Research motivation

Two fundamental issues in real movie recommendation systems are often neglected: scalability and practical usage feedback/verification based on real implementation. Collaborative Filtering (CF) is one of the most widely-used algorithms for making rating predictions within an RS. It is based on the core assumption that users who have expressed similar interests in the past will share common interests in the future. Therefore, the idea of collaborative filtering is to identify users in a community that share appreciation for similar things. Intuitively, if two users have rated the same or almost the same items, then they have similar tastes and can therefore be assigned to a group or close neighborhood. A user can receive recommendations for those items that he/she has not rated before, but that have been already positively rated by users in his/her neighborhood. As the number of users and items grows, CF-based recommendation systems need more resources to process information and form recommendations. The majority of these resources are consumed in determining users with similar tastes and items with similar descriptions. Therefore, CF algorithms face a scalability problem, which can become an important factor for a recommendation system. If the problem is not solved, it is difficult to produce real-time recommendations.

Most existing movie recommendation schemes have worked on improving performance metrics, such as Root-Mean-Square Error (RMSE), utilizing publicly available movie datasets (e.g., MovieLens). The popular methodology is to adopt the so-called 8:2 cross validation, i.e., 80% of the MovieLens data are used as a training set, while the other 20% of data are used for test purposes. However, it is reported that roughly 80% of the publications in this field describe problems or future work that focus on the implementation or verification of systems[5]. This result highlights the importance of these areas in recommendation system development, and shows a need to collect real user feedbacks on movie recommendation through a practically deployed movie recommendation system, and use this real-life data to compare with public dataset-driven research.

1.2 Main contribution

This paper's primary contributions are twofold.

• First, a scalable CF algorithm called Weighted KM-Slope-VU is proposed, which significantly reduces time complexity and is suitable for a real-time recommendation system. Specifically, by exploiting users' profile attributes, our scheme first adopts the K-means clustering algorithm to partition users into several clusters. Each cluster then produces a virtual opinion leader (i.e., a Virtual User (VU)) to represent all other users in the cluster for the evaluation of the items. The Weighted KM-Slope-VU is designed and applied in place of the original user rating data, to reduce the dimensions of the user-item matrix and make a prediction on the basis of the VU-item matrix, which can significantly reduce the time complexity. The method is innovative and efficient; experiments on the MovieLens dataset illustrate that our scheme is comparable to existing work (including two popular Matrix Factorization (MF)-based RS methods, Singular-Value Decomposition (SVD) and SVD++).

• Second, we have constructed a working personalized web-based movie recommendation system called MovieWatch (http://121.42.174.147:8080/Movie/login.action), opened to the public. We collected preliminary feedback from registered users of this live system and used this real-life data as a basis to evaluate the feasibility and accuracy of our system.

The rest of this paper is organized as follows: Section 2 provides a literature survey of previous related works; in Section 3, we present the Slope One algorithm and the newly proposed algorithm—Weighted KM-Slope-VU—in detail, and illustrate its performance through experiments; Section 4 specifies the deployed MovieWatch system, and discusses the experimental results; finally, Section 5 briefly concludes the paper and discusses avenues for future research.

2 Related Work

As shown in Fig. 1, existing recommendation algorithms can be divided into four kinds: content-based, knowledge-based, CF, and hybrid. Among these recommendation algorithms, CF is the most popular technique, based on the core assumption that users who have expressed similar interests in the past will share common interests in the future[5]. CF methods can be model-based or memory-based.

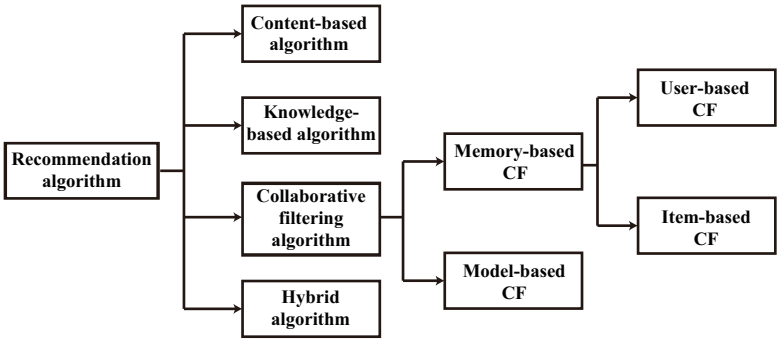


Fig. 1 Categories of existing recommendation algorithms.

Model-based algorithms first construct a model to represent user behavior and, therefore, to predict their ratings. The parameters of the model are estimated using the data from the rating matrix. There are many model-based approaches: Principal Component Analysis (PCA) and SVD are based on algebra[6, 7], Bayes methods are based on statistics[8]. Matrix factorization for recommender systems has been a special focus of a voluminous amount of research, especially since the Netflix Prize competition was announced. This methodology transforms both items and users to the same latent factor space, thus making them directly comparable. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback. For example, when the products are movies, factors might measure obvious dimensions such as comedy vs. drama, amount of action, or orientation to children; less well defined dimensions such as depth of character development or quirkiness; or completely uninterpretable dimensions[9–11]. A prevalent assumption in constructing matrix approximations is that the partially observed matrix is low-rank. LLORMA[12] significantly relaxes this low-rank assumption; instead of assuming that the user-item matrix M can be globally approximated by a low-rank matrix, this method assumes that the matrix M behaves as a low-rank matrix in the vicinity of certain row-column combinations. Therefore several low-rank approximations of M are constructed, each being accurate in a particular region of the matrix. Then a smoothed convex combination of those low-rank matrices is provided to finally approximate the observed matrix as a weighted sum of low-rank matrices. In brief, unlike standard low-rank matrix approximation techniques achieving consistency in the limit of large data (convergence to the data generating process) by assuming that M is low-rank, this local method achieves consistency without the low-rank assumption.

Memory-based approaches can be categorized as user-based or item-based. User-based collaborative filtering algorithms generate recommendations based on the preference of similar users[13, 14]. In contrast to user-based CF, item-based CF approaches recommend items on the basis of information about other items that a user has previously rated[15, 16]. The recommended items for the given user are ranked by the similarities between each candidate item and other items that the user has rated. Since the rating data of each virtual user emerging from a cluster of similar users is used to predict user ratings, our proposed KM-Slope-VU method belongs to the class of user-based CF methods. Reference [17] designs a movie recommendation system using data clustering and computational intelligence, designing an algorithm featuring K-means clustering and cuckoo search optimization, and evaluating the recommendation performance on the MovieLens dataset. Reference [18] develops a hybrid clustering model to improve the movie prediction accuracy and make more pertinent recommendations to users. These authors used a combination of K-means Particle Swarm Optimization (PSO) to find initial centers, which is much more accurate and precise than assigning centers randomly. These centers are then used by a fuzzy C-means for optimization to form the final clusters, which are directly used for the result calculation process. In order to avoid the premature convergence of K-means clustering, Ref. [19] considers a genetic algorithm as the optimization tool for evolving initial seeds in the first step of the K-means process to identify optimal partitions. Experimental results illustrate that their approach is capable of providing more reliable movie recommendations in comparison with the existing

cluster-based CF methods.

However, these recommendation schemes have two common problems. First, based on a user-item rating matrix, K-means is directly employed to find all like-minded users, called neighbors. However, as the number of users and items increases, the dimension of the user-item rating matrix increases quickly, which slows the operation of the clustering algorithm. Second, considering that the sensitivity selection of initial seeds in K-means could influence the final output and easily fall into local optimum, in order to find the nearest neighbor of the user and improve prediction accuracy in RS, K-means often requires a combination of other heavy-computing intelligence algorithms, such as a genetic algorithm like cuckoo search optimization, to form a hybrid clustering model, which also increases the time complexity.

To enable real-time recommendations, it is imperative to solve the scalability problem in a practical movie recommendation system. In response, we propose a simple and scalable KM-Slope-VU recommendation algorithm, with three main features. First, instead of user ratings on items (i.e., user-item matrix), K-means is employed to find several groups of similar users based on their typical profile attributes (age, occupation, etc.), to ensure that the time overhead of clustering does not increase as the number of items increases. Second, each cluster produces a virtual opinion leader (i.e., virtual user) to represent all of the real users in the corresponding cluster, by calculating average rating of the items. Finally, the formed virtual user-item rating matrix is fed into the Weighted Slope One-VU, an improved and lightweight recommendation algorithm based on Slope One, to generate the recommendation results[20]. Compared with the raw user-item matrix, the dimensions of the virtual opinion leader-item matrix are greatly reduced. All of the features above significantly accelerate the classification model training and ensure recommendations can be delivered in real time.

3 Proposed Recommendation Algorithm

Intuitively, we can assume that people who share the same gender, similar age, similar occupations, etc., are likely to share similar tastes in movies, and assign similar ratings to movies. Based on this assumption, we designed a CF algorithm, KM-Slope-VU, in which, according to users' profile attributes, K-means is utilized to partition users into several clusters, and then each cluster produces an opinion leader by calculating the average rating of the items. We also intuit that the historical data of user evaluations of items are naturally correlated to their tastes, and therefore should be utilized to cluster users. Specifically, for each cluster, a virtual opinion leader acts on behalf of other users in the same cluster to rate all items, and then all of virtual opinion leader's ratings are gathered to approximate the real user's rating data. That is, the original user-item rating matrix is replaced with a reduced virtual opinion leader-item rating matrix, meaning that the user-item matrix is compressed and the computational complexity is reduced. Note that the key idea behind the traditional Slope One algorithm is to predict a user's rating for an item based on the score differences between pairs of items. Following this idea, the proposed Weighted Slope One-VU works on the intuitive principle of a popularity differential between items for virtual users in a pairwise fashion, i.e., determines hows much more liked one item is compared to another. One way to measure this differential is to simply subtract the virtual user's average rating of the two items. In turn, this difference can be used to predict real users' ratings of one of those items, given their rating of the other.

In the training phase, we first use K-means to find user neighbors based on profile characteristics, such as age, sex, and profession. Secondly, the average rating of each item is calculated under the same cluster to form a virtual user who represents all users in this cluster. For example, if there are users and movies, that is, the original user-item rating matrix is $R_{m\times n}$, and the K-means outputs K clusters, implying that there are K virtual users, then the virtual user-item rating matrix is $\mathrm{uv}R_{k\times n}$, where uv represents virtual users. Finally, this virtual user-item rating matrix $\mathrm{uv}R_{k\times n}$ is fed into our proposed Weighted Slope One-VU algorithm to infer the recommendation results. In the recommendation phase, our system makes a prediction for users according to the collected user history data.

We use the following notations to describe our scheme. The number of elements in a set S is $\mathrm{card}(S)$; u_i represents the rating given by user u to item i; vu_i denotes the rating given by virtual user vu to item i. The subset consisting of all items rated by the user u is $S(u)$, and $P_{u,i}$ represents the predicted rating of user u on item i.

3.1 Basic Slope One based on virtual user (Slope One-VU)

Given a training set χ including a virtual user's ratings

for all items, the ratings of a virtual user vu $\in S_{i,j}(\chi)$ on items i and j, are vu_i and vu_j, respectively. The average deviation of item i with respect to item j (i.e., the deviation in between the ratings of the virtual user vu on the pair of items i and j) is given as Eq. (1).

$$\text{dev}_{i,j} = \frac{\sum_{\text{vu}\in S_{i,j}(\chi)} (\text{vu}_i - \text{vu}_j)}{\text{card}(S_{i,j}(\chi))} \tag{1}$$

Considering that $\text{dev}_{i,j} + u_j$ is a prediction for u_i given u_j, a reasonable predictor can be the average of all such predictions, as shown as Eq. (2):

$$P_{u,i} = \frac{1}{\text{card}(R_i)} \sum_{j\in R_i} (\text{dev}_{i,j} + u_j) \tag{2}$$

where $R_i = \{j \,|\, j \in S(u), j \neq i, \text{card}(S_{i,j}(\chi)) > 0\}$ is the set of all relevant items.

To illustrate, consider seven users ($A - G$) and two items, as shown in Table 1, with each user having rated items from 1 to 5 (with an empty value meaning that the user has not evaluated that item). We suppose that K-means divides the users into 3 clusters, and each cluster forms a virtual user by calculating the average rating of the items to represent all of the users in the cluster. As shown in Table 2, for example, virtual user 1 gave Item 2 a rating of 4, with the term in parenthesis (2) representing the number of evaluators (in this case, users A and F). The real user H has not yet rated Item 2. We observe that the rating of 4 given by virtual user 1 to Item 2 is 1 point higher than the same vu's rating of Item 1 ($4 - 3 = 1$), while the same differential for virtual users 2 and 3 are -0.5 and 0 points, respectively. Thus we predict that user H will give Item 2 a rating of equation.

3.2 Proposed Weighted Slope One-VU

In the above simple virtual user-based CF methods, each cluster produces a virtual user just by calculating the average evaluation of the items. The weak point of this method lies in the fact that the number of ratings is not taken into consideration. Intuitively, the more times an item is evaluated by real users, the more the rating represents the preferences of other users in the same cluster, and thus the greater the contribution to dev. Thus, we modify Eq. (1) as follows:

$$\text{dev}_{i,j} = \frac{\sum (\text{vu}_i - \text{vu}_j) \cdot \min(\text{vu}_i^n, \text{vu}_j^n)}{\sum_{\text{vu}\in S_{i,j}(\chi)} \min(\text{vu}_i^n, \text{vu}_j^n)} \tag{3}$$

where vu_i^n denotes the number of times that the real user has evaluated the item i in the cluster represented by the virtual user vu. Equation (3) implies that the higher vu_i^n is, the higher the credibility of vu_i should be. When the confidence of vu_i and vu_j are high, the value of $\text{dev}_{i,j}$ is accurate. Therefore, we use $\min(\text{vu}_i^n, \text{vu}_j^n)$ as the weight of $\text{dev}_{i,j}$. As shown in Table 2, we could then predict that user H will give Item 2 a rating of equation.

Table 1 Original user-item rating data (users with same color means they belong to a same cluster).

User	Item 1	Item 2	User	Item 1	Item 2
A	3	4	E	3	3
B	5	4	F	?	4
C	2	3	G	2	3
D	5	?	H	3	?

Table 2 Each cluster producing an opinion leader (virtual user) through calculating the average rating of the items in a same cluster.

User	Item 1	Item 2
Virtual user 1 (A, F)	3(1)	4(2)
Virtual user 2 (B, E)	4(2)	3.5(2)
Virtual user 1 (C, D, G)	3(3)	3(2)
H	3	?

3.3 Experiment using real dataset

3.3.1 MovieLens dataset

To verify the performance of our proposal, we used two real MovieLens datasets: MovieLens 100K and MovieLens 1M; both are collected as part of the GroupLens Research Project of the University of Minnesota (and are available online at https://grouplens.org/datasets/movielens/). The MovieLens 100K dataset consists of 100 000 ratings (with value-5) from 943 users on 1682 movies, and the sparse degree of this dataset is 6.30%. Each user has rated at least 20 movies. The larger MovieLens 1M dataset contains 1 000 209 anonymous ratings of approximately 3900 movies made by 6040 MovieLens users who joined in the year 2000. Both datasets include user demographics.

3.3.2 Evaluation metrics

The most common CF evaluation measure for prediction accuracy is the RMSE, defined as follows:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{u,i\in\text{TestDataset}} (P_{u,i} - r_{u,i})^2} \tag{4}$$

where $P_{u,i}$ and $r_{u,i}$ are the predicted and observed ratings, respectively, for user and item in the test dataset, and N is the number of elements in the dataset. The lower the RMSE is, the more accurate the recommendation engine is at predicting user ratings.

3.3.3 Experimental platform

All of our experiments were implemented using Java and compiled using MyEclipse. We ran the experiments on a PC with an Intel(R) Core(TM) i5-2450M CPU at a speed of 2.50 GHz and with 10 GB of RAM.

3.3.4 Experimental results

For MovieLens 100K, the whole dataset is divided equally into 5 sub-datasets. The first sub-dataset acts as the test data (represented as u1.test), and the other sub-datasets are used as the training data (represented as u1.base). Furthermore, the second sub-dataset is used as u2.test, and the others u2.base. Similarly, we form further training and test sets until we reach u5.test and u5.base. That is, the data sets u1.base and u1.test through to u5.base and u5.test are 80/20 splits of the entire MovieLens 100K dataset into training and test data. Each sub-dataset from u1 to u5 has disjoined test data, adopting the so-called 5-fold cross-validation.

For MovieLens 1M, to illustrate that our proposal is robust for various ways of splitting the dataset, we adopt an alternative splitting method. From the whole dataset, for every consecutive four records, one is randomly selected into the test dataset and the rest are used as training data. That is, the ratio of training set to test set is 3 to 1. Repeating the process five times, we similarly obtain 5 samples of training datasets and test datasets, represented by train1.dat and test1.dat through to train5.dat and test5.dat.

In our experiments, in terms of the metrics of recommendation accuracy and the time complexity, we compare the typical MF-based recommendation algorithms SVD and SVD++[21] with our approach, the KM-Slope-VU and Weighted KM-Slope-VU.

Figures 2 and 3 illustrate the recommendation accuracy of KM-Slope-VU, Weighted KM-Slope-VU, SVD, and SVD++ for MovieLens 100K and MovieLens 1M. We can observe that the average RMSE of our

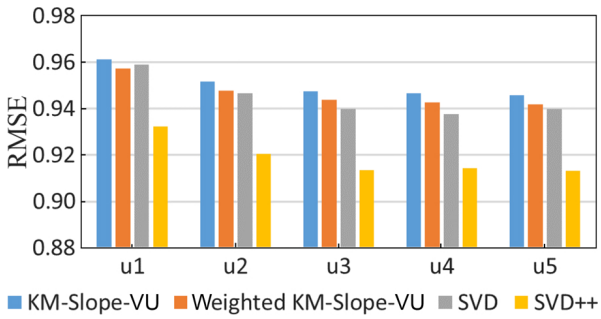


Fig. 2 Comparison of recommendation accuracy (using the dataset of MovieLens 100K).

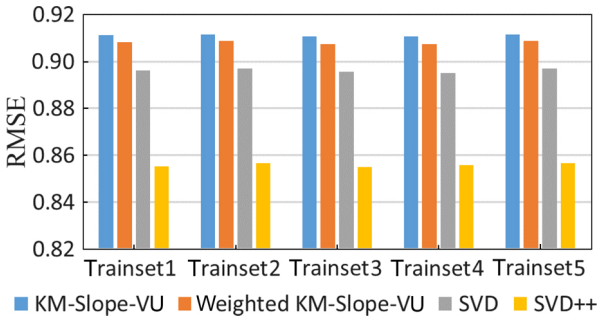


Fig. 3 Comparison of recommendation accuracy (using the dataset of MovieLens 1M).

proposed KM-Slope-VU is 0.950 62 and 0.9111 on the 100K and 1M datasets, respectively. Weighted KM-Slope-VU achieves an average RMSE of 0.946 76 and 0.9081, thus outperforming KM-Slope-VU. The reason for this is that the basic KM-Slope-VU simply averages cluster users' evaluations of items to produce a virtual user for the cluster, without considering the actual number of ratings made by real users. Furthermore, the accuracy of Weighted KM-Slope-VU approximates the average RMSE of SVD, which averages 0.944 64 and 0.896 18 on the 100K and 1M datasets, respectively, while it is inferior to the RMSE of SVD++, which averages 0.9188 and 0.8560. The reason for this is that, to reduce computation complexity, only one VU in a cluster is conceived to represent all of the users in the same cluster, which brings some margin of error and makes our scheme less accurate than SVD++.

Table 3 illustrates the theoretical analysis of time complexity for various recommendation methods. The parameters are explained as follows:

k: number of clusters;

m: number of users;

n: number of items;

t: number of iterations; and

f: number of features/profile attributes of users (such as sex, age, occupation, etc.).

Ratings matrix $R_{m\times n} = P_{m\times d} \times Q_{d\times n}$, where d is the number of factors used in MF-based recommendation methods.

Briefly, the time complexity of KM-Slope-VU can be approximate to $O(m + n^2)$, as the terms k, t, and

Table 3 Comparison of time complexity among different algorithms.

Algorithm	Time complexity
(Weighted) KM-Slope-VU	$kmtf + knn$
SVD	$tmnd$
SVD++	$tmnd$

f can be regarded as small constants relative to m or n. Similarly the computation complexity of MF-based schemes like SVD and SVD++ approximate to $O(m \cdot n)$ (that is, t and d are regarded as the small constants). Usually, the number of rating users in any movie rating platform is much larger than the number of movies ($m \gg n$). Therefore, the (Weighted) KM-Slope-VU performs better than MF in terms of the performance metrics of time complexity. It should be emphasized that in the framework of the KM-Slope-VU recommendation algorithm shown in Fig. 4, K-means clusters users according to the their profile attributes rather than the user-item evaluation data, although the evaluation data of the user directly reflects the user's preferences. A more detailed discussion of this will be presented in Section 4.2.

4 Practically Deployed System and Field Results

Around 80% of publications in the field describe problems or future work needed in the implementation or verification areas[5]. This highlights the importance of these areas to RS development. We developed a live movie recommendation system MovieWatch, in which users were asked to choose a set of their favorite movies, and were then given recommendations for another set of movies (i.e., the "guess what you like" feature). For each recommendation, we request the user for feedback on how good she or he feels about it. At the time of writing this paper, our film site has 134 registered users and has collected 225 pieces of feedback. Since our movie recommendation site is immature, unlike IMDB, Yahoo Movies, etc., we cannot accumulate enough user data in such a short period of time to train our model. Therefore, we utilize the existing user and movie data in the MovieLens dataset, which consists of 100 000 ratings (1 – 5) from 943 users on 1682 movies, and proceed as though the 943 users in this dataset are registered users of MovieWatch. The demographic information for these users includes age, gender, and occupation. The movie genres include action, adventure, animation, comedy, crime, documentary, and 19 other categories, and users can retrieve movie resources via an IMDB URL. Finally, we compare the prediction accuracy between numerical calculations made on the MovieLens dataset and the field experiments conducted on the production MovieWatch environment.

The source code of the MovieWatch prototype system is publicly available on GitHub. The authors welcome all interested scholars to use and test the system at https://github.com/batsqd/Movie.

4.1 MovieWatch personalized movie recommendation site

4.1.1 User registration and login

All users who wish to search through the movie list uploaded by the MovieWatch administrator, and set ratings for movies, are requested first to register with the site. Note that, to protect user privacy, we do not require users to provide any sensitive information (age, gender, identity card number, etc.); only a pseudo-username and password are required. After successful registration, users are able to login into the system,

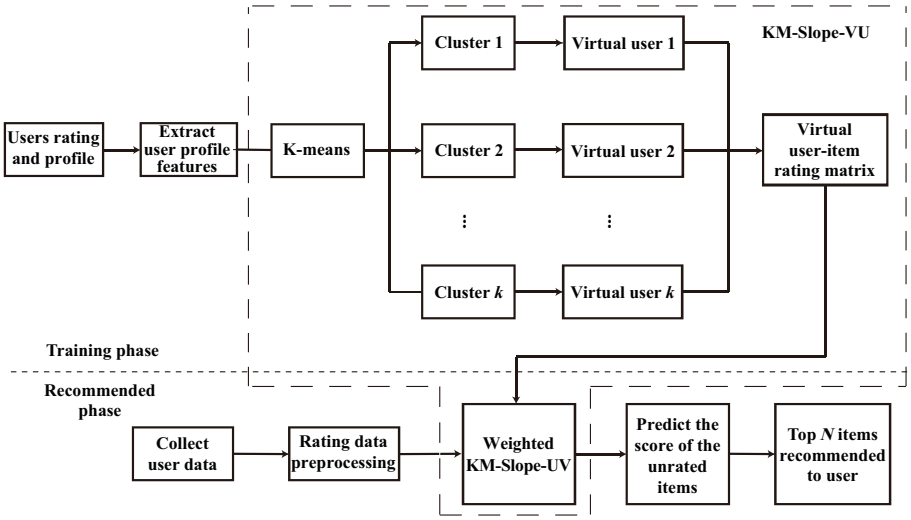


Fig. 4 Framework of the proposed recommendation system.

search for a movie, and rate any movie according to their degree of satisfaction.

4.1.2 User rating for movies

After successful login, a user can directly click on a movie which he/she is interested in to enter the movie details page shown in Fig. 5. From there the user can obtain detailed information about the movie, such as its name, director, main actors, category, release date, rating, synopsis, and a movie player link. The user can also search for a specific movie and rate it, based on a 5-star rating system. If the user gives 5 stars to a movie, this indicates a 100% satisfaction level, and if 1 star is given, the movie does not meet the user's expectations.

4.1.3 Recommendation ("guess what you like")

Recommender systems generally aim to serve a huge number of users, sometimes millions at a time, and there can be very many items to be recommended. Thus, a practical and useful recommender system should have fast real-time processing capabilities. To meet these demands, the MovieWatch system implemented the simple and scalable recommendation method—Weighted KM-Slope-UV—that was proposed in Section 3, providing users with viewing services and recommendations for movies that the user might like. The home page of MovieWatch is shown in Fig. 6.

After entering the home page, the user can select the movie to watch and then rate the movie from 1 to 5 stars. The user can then click "Your Taste" to view

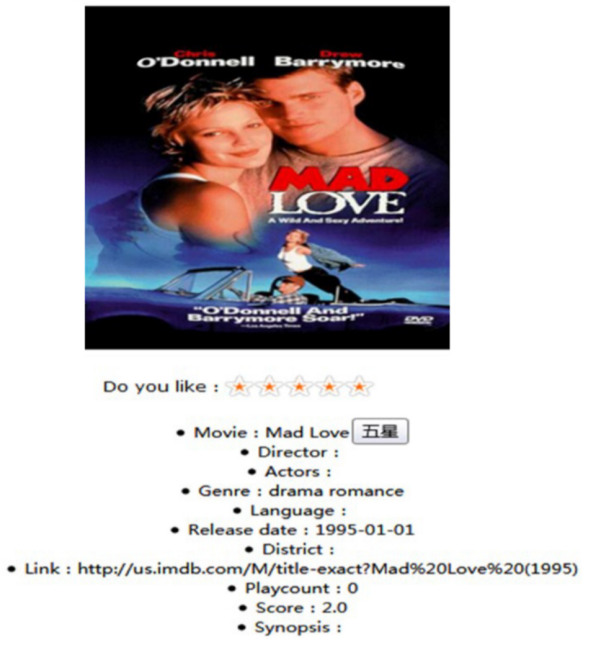


Fig. 5 Screenshot of movie details and rating.

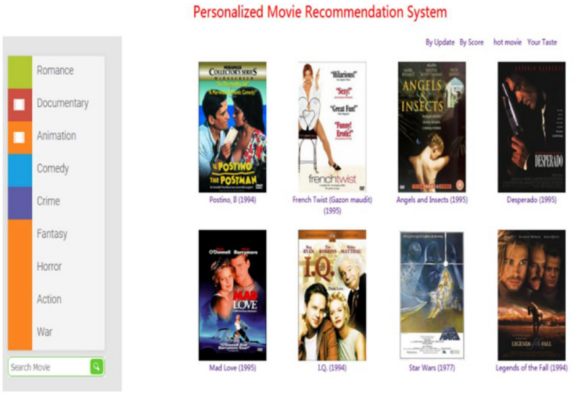


Fig. 6 Home page of MovieWatch.

the list of movies recommended by the MovieWatch system. Note that, in its current state, the personalized recommendation movie site MovieWatch is best suited to the Google Chrome and Firefox web browsers.

4.1.4 Validation in production environment

We evaluated the accuracy of our algorithm, Weighted-Slope-UV in the live production environment. So far, we have attracted 137 registered users and collected 225 pieces of feedback with users' opinions on the movies recommended by the MovieWatch system. We evaluate the effectiveness of our proposed algorithm based on the actual difference between the prediction of the algorithm and the real feedback rating data.

Figure 7 shows that the numbers of users who gave recommended movies a rating from 1-star to 5-star are 6, 16, 49, 101, and 53, respectively. Note that MovieWatch only recommends to users movies with a predicted rating higher than 3 stars (inferred by our proposed Weight KM-Slope-UV). Therefore, intuitively, the user satisfaction level of MovieWatch recommendations is 90.2% (i.e., 203/225). In contrast to the popular Top-N RS systems, in which the

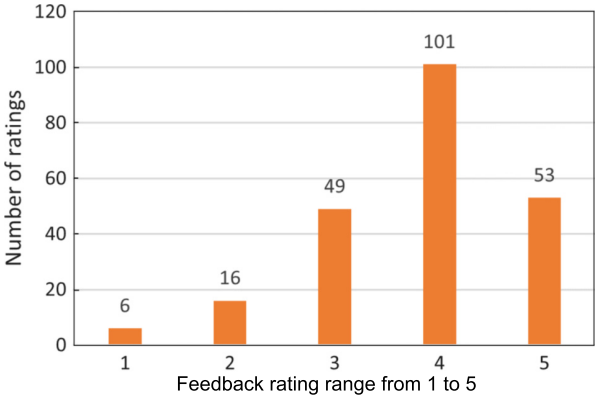


Fig. 7 Distribution of feedback ratings.

top 5 movies with the highest predicted score are recommended to each user, for its list of recommendations, the MovieWatch system randomly selects 5 movies out of those that have a predicted rating higher than 3 stars. The reason for random selection is that, considering that taste is somewhat subjective and has an intrinsic uncertainty, a fuzzy process can better incorporate the subjective dispositions of users, and in turn enhance users' satisfaction with the movie recommendations. Furthermore, each time a user clicks Your Taste, he or she can obtain similar but slightly different results as recommended candidates. This is a more user friendly method for a practical movie recommendation system.

Figure 8 illustrates the prediction errors (in terms of RMSE) of our proposed KM-Slope-VU and Weighted KM-Slope-VU, when they are conducted on the MovieLens dataset (the right two bars), and the RMSE of Weighted KM-Slope-VU (the left bar) on user feedback collected from our deployed MovieWatch system. We can observe that, in a live production environment, the average RMSE of Weighted KM-Slope-VU reaches 1.084, slightly worse than the performance value of 0.946 76 obtained when conducting Weighted KM-Slope-VU on MovieLens dataset (using the popular 8:2 cross validation methodology, as explained above). The reason for this result is given in the following discussion subsection.

4.2 Discussion

As a matter of fact, almost all of the registered users of MovieWatch are students of Nanjing University of Posts and Telecommunications, and most are within the 18 to 25 age bracket. Figures 9 – 11 illustrate the demographic information of the users in the MovieLens 100K dataset in terms of gender, age,

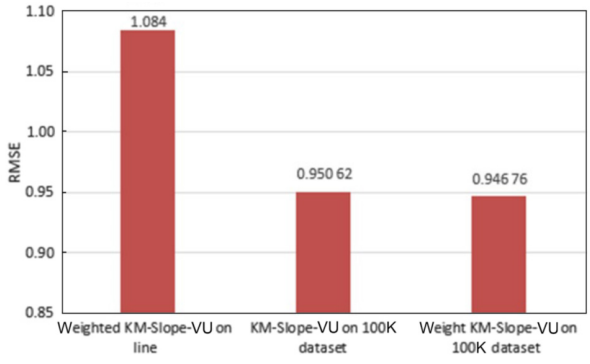


Fig. 8 Comparison of our scheme using the data collected by MovieWatch and datasets of MovieLens.

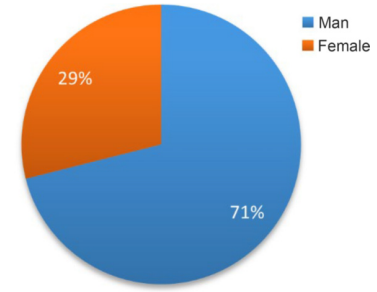


Fig. 9 Gender ratio of users in MovieLens dataset.

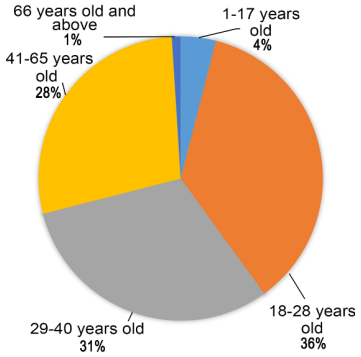


Fig. 10 Age distribution of users in MovieLens dataset.

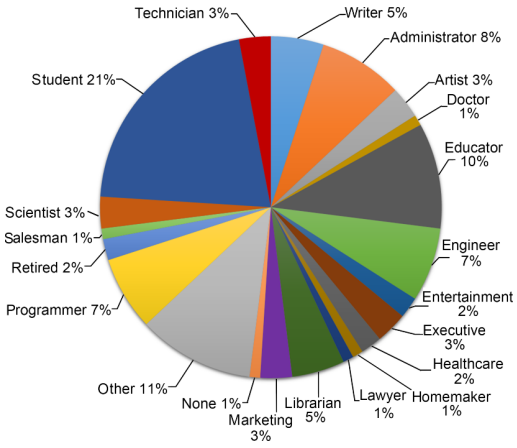


Fig. 11 Occupational distribution of users in MovieLens dataset.

and occupation, respectively. Between the registered users in our MovieWatch system and the users in the MovieLens datasets, there are significant differences in age and occupation. To alleviate the bootstrapping problem that any new movie recommendation system faces, we mixed MovieLens 100K data with user feedback data collected online to train the model and then used that model to predict the rating of registered users. However, the difference in demographics may have led to a deviation in prediction accuracy between the MovieWatch system and the MovieLens dataset. Moreover, most movies in the MovieLens dataset were

produced twenty years ago, and the younger users of MovieWatch may not be interested in the outdated movies in the dataset, which might be another reason for the gap in recommendation accuracy between those two data environments.

There are two underlying reasons why, instead of clustering users directly based on the user-item matrix, our scheme intentionally clusters users according to their profile attributes. First, the computational overhead of clustering on users' profile attributes is significantly less than that of clustering on the raw user-item matrix. Second, it is sometimes difficult or even impossible to conduct clustering using the raw matrix due to its sparseness. This can be seen in the example shown as Table 4, in which users 1 and 2 have no common ratings; it is impossible to infer similarity between those two users, thus no clustering can take place. We have run our proposed Weighted KM-Slope-VU on the MovieLens 100K and 1M datasets using clustering based on raw matrix and simple user profile attributes and, as shown in Figs. 12 and 13, respectively, comparable RMSEs are achieved.

Finally, in our scheme, Weighted KM-Slope-VU, the popular K-means algorithm is chosen to cluster users, for its simplicity and effectiveness. The contingent issue is how to practically determine the proper number of clusters, i.e., the value of K. This problem can be partially solved as follows. Depending on the specific recommendation system, there typically exist several

Table 4 Illustration of 2 users rating on 7 movies.

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6	Movie 7
User 1	3	3	3	-	-	-	-
User 2	-	-	-	4	4	4	-

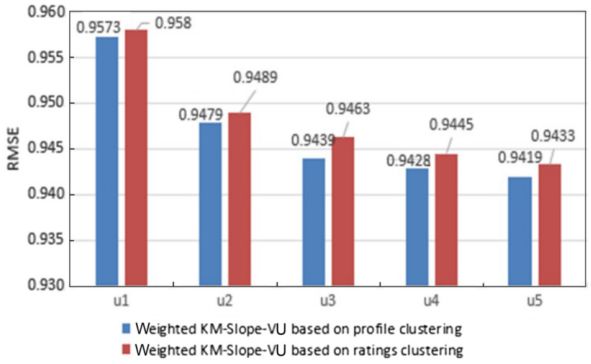


Fig. 12 Comparison of the accuracy of KM-Slope-VU based on user profile and raw rating matrix (MovieLens 100K dataset).

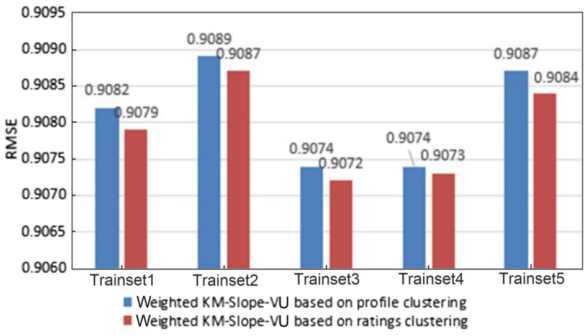


Fig. 13 Comparison of the accuracy of KM-Slope-VU based on user profile and raw rating matrix (MovieLens 1M dataset).

key latent factors/user profile attributes (sex, age, occupation, etc.), and each attribute can be further divided into several categories. For instance, the number of sex categories is 2, age categories might be 3, and main occupation categories might be 5. Using these figures, the total number of categories can be roughly estimated as 30 (i.e., $2\times3\times5$). As it happens, in our experiments, setting K to 30 can achieve good performance.

5 Conclusion and Future Work

In this paper, we developed a novel collaborative filtering approach called Weighted KM-Slope-VU for fast and scalable movie recommendations, and furthermore developed and deployed a personalized movie recommendation site, MovieWatch, to provide users with viewing services and collect user feedback on recommended movies to practically evaluate our proposed algorithm using real-life data. Specifically, we adopted K-means to partition users into several clusters, and then for each cluster conceived a virtual opinion leader to represent all of the users in that cluster. Then, instead of processing the original full user-item rating matrix, a reduced virtual opinion leader-item matrix is processed by the proposed Weighted Slope One-VU recommendation algorithm. Experiments on MovieLens datasets show that our scheme can achieve performance (measured by RMSE) comparable with recommendation algorithms based on matrix factorization, but reduce time complexity in common scenarios. Furthermore, a practical movie recommendation system called MovieWatch was developed, deployed, and opened to the public to collect user feedback on the movies recommended to them. Our scheme was then evaluated based on this real

feedback by registered users of MovieWatch.

However, in the present work, only one virtual opinion leader is conceived to represent each whole cluster. Because this may lead to the loss of valuable information in the cluster, the prediction accuracy of the proposed algorithm is slightly lower than that of SVD and SVD++. Furthermore, due to some outdated movies and the small number of data samples collected by our MovieWatch system, the real feedback evaluation is somewhat higher than the performance obtained on the MovieLens dataset. For future work, we plan to improve our recommendation system in the following two ways: first, make the latest films available for users instead of having outdated movies in the dataset; and second, optimize the selection of virtual users to better represent the entire set of real users in a cluster, by using for instance fuzzy C-means[22], which could further improve the recommendation accuracy.

References

[1] C. G. Chiru, C. Preda, V. N. Dinu, and M. Macri, Movie recommender system using the user's psychological profile, in *IEEE International Conference on Intelligent Computer Communication and Processing*, Cluj-Napoca, Romania, 2015.

[2] M. N. Jelassi, S. B. Yahia, and E. M. Nguifo, A personalized recommender system based on users' information in folksonomies, in *Proc. 22nd Int. Conf. World Wide Web*, Rio de Janeiro, Brazil, 2013.

[3] X. B. Wang, F. J. Luo, C. Y. Sang, J. Zeng, and S. Hirokawa, Personalized movie recommendation system based on support vector machine and improved particle swarm optimization, *IEICE Trans. Inf. Syst.*, vol. E100.D, no. 2, pp. 285–293, 2017.

[4] H. Li, J. T. Cui, B. Q. Shen, and J. F. Ma, An intelligent movie recommendation system through group-level sentiment analysis in microblogs, *Neurocomputing*, vol. 210, pp. 164–173, 2016.

[5] I. Portugal, P. Alencar, and D. Cowan, The use of machine learning algorithms in recommender systems: A systematic review, *Expert Syst. Appl.*, vol. 97, pp. 205–227, 2018.

[6] D. Goldberg, D. A. Nichols, B. M. Oki, and D. Terry, Using collaborative filtering to weave an information tapestry, *Commun. ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, Incremental SVD-based algorithms for highly scalable recommender systems, in *Proc. 5th Int. Conf. Computer and Information Technology*, Dhaka, Bangladesh, 2002.

[8] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, Eigentaste: A constant time collaborative filtering algorithm, *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[9] J. S. Breese, D. Heckerman, and C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in *Proc. 14th Conf. Uncertainty in Artificial Intelligence*, Madison, WI, USA, 2013, pp. 43–52.

[10] X. Y. Su and T. M. Khoshgoftaar, A survey of collaborative filtering techniques. *Adv. Artif. Intell.*, vol. 2009, p. 4, 2009.

[11] Y. Shi, M. Larson, and A. Hanjalic, Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges, *ACM Comput. Surv.*, vol. 47, no. 1, pp. 3, 2014.

[12] Y. Koren, Factorization meets the neighborhood: A multifaceted collaborative filtering model, in *Proc. 14th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Las Vegas, NV, USA, 2008, pp. 426–434.

[13] J. Lee, S. Kim, G. Lebanon, and Y. Singer, Local low-rank matrix approximation, in *Proc. 30th Int. Conf. Machine Learning*, Atlanta, GA, USA, 2013.

[14] J. Herlocker, J. A. Konstan, and J. Riedl, An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms, *Information Retrieval*, vol. 5, no. 4, pp. 287–310, 2002.

[15] R. Jin, J. Y. Chai, and L. Si, An automatic weighting scheme for collaborative filtering, in *Proc. 27th Annu. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Sheffield, UK, 2004, pp. 337–344.

[16] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, Item-based collaborative filtering recommendation algorithms, in *Proc. 10th Int. Conf. World Wide Web*, Hong Kong, China, 2001.

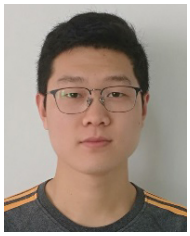
[17] M. Deshpande and G. Karypis, Item-based top-N recommendation algorithms, *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, 2004.

[18] R. Katarya and O. P. Verma, An effective collaborative movie recommender system with cuckoo search, *Egypt. Inf. J.*, vol. 18, no. 2, pp. 105–112, 2017.

[19] R. Katarya and O. P. Verma, A collaborative recommender system enhanced with particle swarm optimization technique, *Multimed. Tools Appl.*, vol. 75, no. 15, pp. 9225–9239, 2016.

[20] Z. Wang, X. Yu, N. Feng, and Z. H. Wang, An improved collaborative movie recommendation system using computational intelligence, *J. Visual Lang. Comput.*, vol. 25, no. 6, pp. 667–675, 2014.

[21] D. Lemire and A. Maclachlan, Slope one predictors for online rating-based collaborative filtering, in *Proc. 2005 SIAM Int. Conf. Data Mining*, Newport Beach, CA, USA, 2005, pp. 21–23.



Jiang Zhang a master student of Nanjing University of Posts and Telecommunications (NUPT), China. His research interest is big data-based recommendation system.

Zhiyuan Yuan received the MS degree from Nanjing University of Posts and Telecommunications (NUPT), China in 2018. His research interest is big data-based recommendation system.

Qun Jin is a full professor at the Networked Information Systems Laboratory, Department of Human Informatics and Cognitive Sciences, Faculty of Human Sciences, Waseda University, Japan. He has been extensively engaged in research works in the fields of computer science, information systems, and social and human informatics. He seeks to exploit the rich interdependence between theory and practice in his work with interdisciplinary and integrated approaches. His recent research interests cover human-centric ubiquitous computing, behavior and cognitive informatics, big data, data quality assurance and sustainable use, personal analytics and individual modeling, intelligence computing, blockchain, cyber security, cyber-enabled applications in healthcare, and computing for well-being. He is a senior member of ACM, IEEE, and Information Processing Society of Japan (IPSJ).

Yufeng Wang received the PhD degree from Beijing University of Posts and Telecommunications (BUPT), China. He acts as a full professor in Nanjing University of Posts and Telecommunications, China. From March 2008 to April 2011, he acted as an expert researcher in National Institute of Information and Communications Technology (NICT), Japan. His research interests focus on cyber-physical-social systems, mobile social networks, etc.