Mining Item Popularity for Recommender Systems

Jilian Zhang[1], Xiaofeng Zhu[1], Xianxian Li[1], and Shichao Zhang[1,2,⋆]

[1] College of CS & IT, Guangxi Normal University, Guilin, 541004, China
[2] The Centre for QCIS, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia
{zhangjl,zhuxf,lxx,zhangsc}@mailbox.gxnu.edu.cn

Abstract. Recommender systems can predict individual user's preference (individual rating) on items by examining similar items' popularity or similar users' taste. However, these systems cannot tell item's long-term *popularity*. In this paper, we propose an algorithm for predicting item's long-term popularity through *influential* users, whose opinions or preferences strongly affect that of the other users. Consequently, choices made by certain influential users have the tendency to steer subsequent choices of other users, hence setting the popularity trend of the product. In our algorithm, specifically, through judicious segmentation of the rating stream of an item, we are able to determine whether it is popular, and whether that is the consequence of certain influential users' ratings. Next, by postulating that similar items share similar influential users, and that users rate similar items consistently, we are able to predict the influential users for a new item, and hence the popularity trend of the new item. Finally, we conduct extensive experiments on large movie rating datasets to show the effectiveness of our algorithm.

Keywords: Item Popularity, Recommender System, Collaborative Filtering.

1 Introduction

The easy accessibility of recommendations or comments from experts, friends, colleagues, and famous websites is exerting an increasingly greater influence on consumers' purchasing behavior [3] in the era of web 2.0 [7,10,11]. People would pour through detailed reviews written by professional and casual users on C|net or ZDnet, before deciding on a certain brand or model of electronic camera or laptop. Others surf IMDB.com or Amazon.com for user opinions to determine whether a newly released movie or a new book is worth buying. By studying how certain published opinions are likely to impact the appeal of a product to particular groups of consumers, a retailer could gain valuable insights for formulating his sales strategy. The problem of relating past recommendation to future user preference has been studied extensively in the context of collaborative filtering (CF for short) recommender system, which is an appealing research topic in Web Intelligence. CF methods [1,16] have been proposed to address the problem and some of them have been deployed successfully by commercial websites such as eBay, eLance and Amazon. Specifically, the CF algorithms are used to solve this kind of prediction problem: given a user preference database on a set of

⋆ Corresponding author.

H. Motoda et al. (Eds.): ADMA 2013, Part II, LNAI 8347, pp. 372–383, 2013.

items, one wants to predict a specific user A's preference (in the form of rating) for an item x. This task can be tackled using (1) item-based methods, which predict A's rating on x by analyzing A's past preferences on other items similar to x, and (2) user-based methods, which estimate the rating result based on preferences on x expressed by other users who have taste similar to that of A. Unfortunately, those methods are designed to predict individual preference that is in the form of a single rating number, resulting in the fact that there has been little effort on modeling long term popularity of items. And the predicting power of influence users, whose opinions or preferences strongly affect that of the other users, has been neglected by existing CF methods.

The goal of this paper goes beyond individual user ratings, to predicting the popularity of an item across several users over a period of time through influential users. We draw inspiration from research on statistical herding, contagion, and information cascades, which provide evidence that the current demand for a product depends on public information about its past demand [13]. Consequently, the choices made by certain influential users have the tendency to steer subsequent choices of other users, hence setting the popularity trend of the product. For example, by identifying influential users who are likely to rate a product favorably, a merchant could target them early in his promotion campaign, and adjust the product pricing according to their reviews.

In this paper, we propose a method for identifying influential users, which are then utilized for predicting popular items based on past user reviews, in the form of historical rating data. Specifically, we firstly identify popular items from historical rating data through judicious segmentation of item rating stream; next statistical test- t test, is performed on changes of item's rating sequence so that we can determine which user is an influential user; and then by postulating that similar items share the same influential users, and that users rate similar items consistently, we are able to predict the influential users for a new item, the ratings that they are likely to assign, and hence the popularity trend of the new item.

2 Preliminaries

2.1 Related Work

The common goal of data mining techniques in recommender systems is to improve the quality of recommendation through text mining or summarization of online review comments. Hu and Liu's work in [6] apply text mining schemes to summarize customer review data on the Web. In a departure from conventional text summarization, their algorithm generates product features only from review sentences that express the opinions of the writers, and it can identify whether each opinion is positive or negative. The outputs at the end are useful for customers to decide whether to purchase a product, and for manufacturers to track and manage customer opinion on their products. A similar work was conducted by Archak et al [2], in which a hybrid technique combining text mining and econometrics is proposed to derive the quantitative impact of consumer textual reviews on products as a linear function with the help of tensor product technique, so that the pricing power of a product can be computed from the consumer review data. There are also some examples of exploiting both textual and numeric review data in order to make a better recommendation. Different from the above methods, our algorithm takes into account consumer's numeric comment data (i.e., ratings for product), and it can efficiently identify those users who may exert potential influence on the others and predict future popularity of products.

Extensive studies on recommendation systems have focused on collaborative filtering (CF) of user-item rating data in. Existing CF schemes fall into two main categories: model-based and memory-based approaches. For the model-based approach, a model learned from a training dataset is used to estimate the ratings for active users from prior user ratings. Clustering smoothing model [9,14], aspect model [5], and Bayesian network [15] exemplify this line of work. In contrast, memory-based approaches perform calculations on the entire rating dataset in order to find the K most similar users (or items) to the active user (or item) with respect to some similarity measures, then combine the corresponding ratings of these similar users (or items) by using simple weighted sum or regression [12]. Sarwar et al demonstrated that the item-based method greatly outperforms the user-based method in terms of running time, quality of recommendation, and scalability [12].

As discussed above, traditional CF algorithms have achieved tremendous success in recommender systems, with many novel extensions appeared. However, they only focus on how to make a better prediction for an individual rating through the help of various models that they build using machine learning and statistic techniques. As another new extension of the standard CF algorithm, our algorithm goes beyond predicting individual rating for items, to forecasting long-term popularity of item. We propose the concepts of *influential user* and *popular item* in the context of collaborative filtering, and devise efficient algorithm to identify influential user and popular item, which are combined to predict the long-term popularity of new items.

2.2 Problem Definition

While collaborative filtering (CF) has been employed successfully for personalized recommendation in real-world applications, existing CF methods focus only on predicting individual ratings for active users, i.e., individual users' preference for selected items. However, it would be very valuable to go beyond that, to predicting the long-term *popularity* of an item (Note that the definitions of popularity and influential user will be given in the next section). For example, an online bookstore would like to know whether a newly launched book will be popular with its existing customers, in order to devise an appropriate promotion campaign. In real life, products become popular for a variety of reasons. Obvious ones include high quality and consistent performance. There may also be factors that extend beyond any inherent characteristics of the product. In particular, there is evidence that oftentimes the expressed views of a group of customers (or a single customer) have enough clout to steer the preference of subsequent customers. Such customers are called *influential users*. For example, a new book with highly positive reviews (or ratings) by famous critics has a high probability of enjoying brisk sales. We define the problem of predicting an item's popularity as follows

Given a user-item matrix A with rating time for each individual rating and an item I, find a set of popular items x and a group of influential users u from A, and then predict I's popularity based on A, x, and u.

3 The Predicting Model

In this section, we introduce an algorithm for predicting the long-term popularity of an item. The algorithm is centered on the instrumental role that influential users play in shaping the popularity trend of the item. As summarized in Algorithm 1, the algorithm consists of three major stages, including (1) forming clusters of similar terms from the user-item matrix, then identifying the popular items within each cluster (lines 2-5); (2) identifying the influential users who are likely to be responsible for the popular items in each cluster, as well as characterizing the influence of various ratings from those users (lines 6-8); and (3) combining the popularity trends of similar items with the influence exerted by the influential users, into a prediction of the active item's popularity (line 9-16). The following sections elaborate on each stage of the algorithm.

Algorithm 1. Predicting Item Popularity

Input: user-item rating matrix A, an active item X, $\delta, \theta, \lambda, K$
Output: popularity of X
1 Generate rating sequences Q from A; and cluster the items in Q;
2 **for** *each item I in cluster C* **do**
3   Segment item I's rating sequence;
4   Compute the popularity of I;
5   Identify candidate influential users of I;
6 Compute the top-K influential users for each cluster;
7 **for** *each top-K influential user u in cluster C* **do**
8   Compute the influence of u in C under different ratings;
9 **for** *each cluster C* **do**
10   **for** *each item I in C* **do**
11   Compute the similarity between X and I;
12   Compute X's prior popularity $\tilde{H}_C(X|r)$;
13   **for** *each top-K influential user u in C* **do**
14   Predict u's rating for X;
15   Compute X's within-cluster popularity $\hat{H}_C(X)$ in C;
16   Compute X's overall popularity $\hat{H}(X)$;
17 Return $\hat{H}(X)$;

3.1 Identify Popular Items from User Rating Data

The rating sequence R_i of an item i is denoted as $R_i = R_{i1}, R_{i2}, ..., R_{im}$, where m is the number of users. R_{i1} is the rating given by the first user and R_{im} is the score assigned by the latest user. Figure 1 illustrates the transformation of user-item rating matrix to item rating sequences according to the time that user gave rating to items. To find popular items, we apply time series analysis to segment the rating sequences, and look for those that experience sharp spikes in ratings and/or prolonged periods of above-average ratings relative to other similar items.

	I_1		I_2		I_3		I_4	
	Rating	Rating time	Rating	Rating time	Rating	Rating time	Rating	Rating time
U_1	3	09:58	2	11:00	5	20:35	5	10:02
U_2	8	09:12	3	21:45	6	15:06	8	09:22
U_3	5	17:05	1	08:10	4	18:37	2	09:05
U_4	2	11:40	5	10:30	8	22:41	7	17:38
U_5	4	13:12	3	08:12	5	23:36	4	12:10
U_6	6	09:42	7	19:20	3	14:10	5	19:10

An artificial example of user-item rating matrix with rating time

	I_1
U_2	8
U_6	6
U_1	3
U_4	2
U_5	4
U_3	5

	I_2
U_3	1
U_5	3
U_4	5
U_1	2
U_6	7
U_2	3

	I_3
U_6	3
U_2	6
U_3	4
U_1	5
U_4	8
U_5	5

	I_4
U_3	2
U_2	8
U_1	5
U_5	4
U_4	7
U_6	5

Rating sequences sorted by rating time for item I_1, I_2, I_3, and I_4

Fig. 1. An example of how to generate item rating sequence

Segment the Rating Sequence of an Item. The first step in analyzing a rating sequence is to segment it. We adopt the bottom-up Piecewise Linear Representation (PLR) method with slight modification. PLR is a common technique for time series segmentation. PLR variants include sliding windows, top-down, and bottom-up. While the sliding windows has the lowest time complexity and relatively low representation quality, the bottom-up method is able to achieve very good segmentation results with only slightly higher cost [8]. We therefore follow the bottom-up method. In our algorithm, an item's rating sequence is first divided into $m/2$ equal segments, where m is the total number of users who have rated this item. Consecutive segments are then merged iteratively. The merging criterion uses the t-statistic to test whether the difference in mean ratings of two adjacent segments i and j is larger than some threshold δ at some confidence level α. Then the t statistic is compared with the t distribution to determine whether the hypothesis, i.e., the difference in mean ratings is larger than δ, should be rejected [4]. If so, the segments cannot be merged; otherwise segments i and j are merged to form a longer segment. The process is repeated until no adjacent segments can be merged. An example in Figure 2 illustrates how the rating sequence for an item x is segmented.

Compute Item Popularity. Intuitively, the popularity of an item should be judged against other items with similar adoption patterns. For example, it is not useful to compare an educational documentary, which experiences slow adoption over a long life cycle, with the latest mobile phone model that has only a short time span to capture consumers' attention. We therefore cluster the items by the similarity of their rating sequences as advocated in [14].

The similarity measure for the clustering step is the Adjusted Cosine Similarity (ACS), which is widely used for computing item-item similarity. Depending on the intended use of the predicted output, a popular item may be one that receives above average user ratings over prolonged intervals, or substantially higher than average user ratings even if over only a short period. The former suits a merchant who is looking for steady returns, whereas the latter is useful if the merchant is ready to capitalize on demand spikes. The *popularity* of an item x is thus defined as

$$H_c(x) = \lambda \frac{\sum_{l \in S_x} E_{xl}}{\sum_{i \in C} \sum_{l \in S_i} E_{il}} + (1-\lambda) \frac{\sum_{l \in S_x} L_{xl}}{\sum_{i \in C} \sum_{l \in S_i} L_{il}} \tag{1}$$

where E_{xl} and L_{xl} denote the mean rating and length of segment l of item x, respectively; S_x is the set of x's segments with mean segment rating greater than the mean of

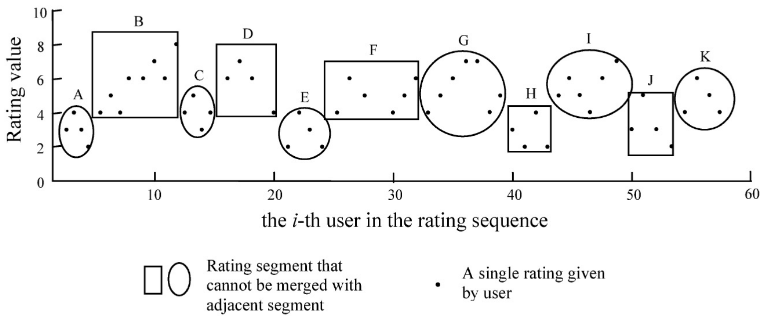


Fig. 2. Segmentation result of an item's rating sequence

the entire rating sequence, i.e., $E_{xl} \geq E_x$ for all $l \in S_x$. $\lambda (0 \leq \lambda \leq 1)$ is a parameter for tuning the relative importance of E_{xl} versus L_{xl}. If $\lambda = 1$, the item popularity relies completely on how high the mean segment ratings of S_x is; if $\lambda = 0$, the item popularity is determined only by the length of the segments in S_x. A higher $H_c(x)$ means that item X is more popular than the others in a same cluster.

3.2 Identify Influential Users Based on Item Popularity

Based on previous research [13], influential users are those whose ratings on an item give a significant boost to the preference of subsequent users. Referring to Figure 2, suppose that D is a high-rating segment. A high-rating segment is one that has a higher mean rating than the item's overall average rating. If the mean of the high-rating segments after D is significantly greater than that of the high-rating segments before D, then the users who contributed to segment D are candidate influential users who potentially are responsible for the improved ratings. The influence levels extended by various rating assignments from the K most influential users are then characterized.

Find Candidate Influential Users. To find the *C*andidate *INF*luential *U*sers (*CINFU*), we perform the following procedure starting with the first segment of each item. Suppose that, in Figure 2, B, D, F, G, and I are the high-rating segments. In processing D, we use the t-test to check whether the average of F, G, and I is significantly greater than that of B. If the difference is significant, we claim that the users who contributed to D are candidate influential users. The test then proceeds to each of the high-rating segments after D, until H_0 is rejected.

Identify Top-K Influential Users. While the above procedure produces the *CINFUs*, we cannot use them directly for predicting popular items. The reasons are: 1) Some *CINFUs* may have strong influence on many items, especially on many popular items, whereas other *CINFUs* may exert only limited influence on a few items. 2) The *CINFU* set may contain false positives, i.e., general users who have low influence on future user preferences. 3) There may be hundreds or even thousands of *CINFUs* for just one cluster of items; obviously, this will degrade the efficiency of our prediction algorithm.

To identify the genuine influential users, a *CINFU* u in cluster C is defined as

$$Score(u, C) = \sum_{x \in C \wedge R_{u,x} \neq 0} \frac{|\overline{R}_B(x) - \overline{R}_A(x)|}{R_{max}} * H_C(x) \tag{2}$$

where x is an item that has been rated by this *CINFU*; $\overline{R}_B(x)$ and $\overline{R}_A(x)$ are the mean ratings of the high-rating segments before and after this *CINFU* in x's rating sequence; R_{max} is the maximum possible rating; and $H_C(x)$ is the popularity of item x in C as defined in Equation 1. The scoring function recognizes u to be an influential user if u has rated many items, and most of those items receive higher mean ratings after u.

Finally, the top-K influential users U_K^C for each cluster C, are the K *CINFUs* in C who possess the largest scores:

$$U_K^C = \{u | Rank(Score(u, C)) \leq K\} \tag{3}$$

This selection process is intended to weed out the false positives, as well as to improve computational efficiency.

Derive the Influence of Top-K Influential Users. Among the influential users (denoted as *INFU*) in U_K^C, the influence exerted by each may still vary greatly. Even for a given *INFU*, the impact of his rating on an item could be conditioned on the actual rating value. For example, a poor rating from a particular *INFU* could sink the popularity of an item, but a favorable rating from the same *INFU* may not have the opposite effect of raising the item's popularity. The converse may also be true. Hence, we need to characterize the influence for each *INFU* for different ratings.

The following formula quantifies the influence of *INFU* on the popularity of an item in cluster C, conditioned on the rating value:

$$INF(u, r) = \frac{\sum_{j \in C} \{H_C(j) * I(R_{u,j}, r)\}}{\sum_{j \in C} H_C(j)} \tag{4}$$

where $I(.)$ is an indicator function such that $I(R_{u,j}, r) = 1$ if $R_{u,j} = r$, otherwise 0; $INF(u, r)$ is the *influence* of *INFU* u across the items in cluster C when u gives rating r. $INF(u, r)$ ranges from 0 to 1 by definition.

3.3 Predict the Popularity of a New Item

In collaborative filtering, individual ratings for an active user are usually predicted with a trained model (in the model-based approach) or other memory-based methods. In contrast, in this paper the predicted popularity trend of active item is derived from our proposed framework.

We observe that many products follow certain familiar adoption patterns. For example, electronic gadgets like mobile phones tend to command consumers' attention when they are launched, but the interest falls quickly over time. In contrast, big ticket items like medical equipments and enterprise software take time to gain acceptance. The existence of common adoption patterns imply that the active item's popularity should bear close semblance to those of similar items. This observation can be harnessed to predict the popularity of the active item.

Given an active item X, we first determine its prior popularity from each cluster, conditioned upon X receiving rating r. Next, we predict how the influential users of each cluster are likely to rate X, and arrive at the popularity of X judging from that cluster. Finally, summing over the most similar clusters gives X's predicted popularity.

Prior Popularity: The *prior popularity* of item X in cluster C, conditioned upon X being rated r, is:

$$\tilde{H}_C(X|r) = \frac{\sum_{j \in V_r} Sim(X, j) * (H_C(j) - \overline{H}_C(V_r))}{\sum_{j \in V_r} |Sim(X, j)|} \tag{5}$$

Here $Sim(X, j)$ is the ACS similarity between item X and j, V_r is the set of items each of which is rated r by any of the *INFUs* in U_K^C; and $\overline{H}_C(V_r)$ is the mean popularity of all the items in V_r .

Within-cluster Popularity: Next, combining the prior popularities for different r ratings, weighted by the corresponding influence of the *INFUs*, gives X's popularity with respect to each cluster. Specifically, the *within-cluster popularity* of item X in C, $\hat{H}_C(X)$, is the product of the *prior popularity* and the predicted ratings from *INFU* (as given by Equation 4):

$$\hat{H}_C(X) = \sum_r \left(\sum_{u \in U_K^C} INF'(u, r) * \tilde{H}_C(X|r) \right) \tag{6}$$

In the above formula, we need to normalize the influence of *INFUs* in conditioned upon X being rated r, so as to ensure that the *influence* of the *INFUs* are weights that add up to 1. Therefore, we have $INF'(u_j, r) = INF(u_j, r) / \sum_{u_i \in U_K^C} INF(u_i, r)$. An example of how to compute the *within-cluster popularity* is given below.

Overall Popularity: With Equation 6, we could sum up $\hat{H}_C(X)$ over all possible clusters C to arrive at the predicted popularity of X. In practice, those clusters that are similar to X are expected to account for most of the influence on X's popularity. To reduce computation cost, we compute the final *popularity* of X only from C_N, the N nearest clusters of X. Specifically, the overall popularity of X over C_N is the sum of each cluster C's $(C \in C_N)$ *within-cluster popularities*, weighted by the similarity between X and C's centroid:

$$\hat{H}(X) = \frac{\sum_{c \in C_N} Sim(X, C) * \hat{H}_C(X)}{\sum_{c \in C_N} |Sim(X, C)|} \tag{7}$$

4 Empirical Evaluation

In this section, we present a set of comprehensive experiments to study the effectiveness of our proposed framework.

Dataset. We use two MovieLens datasets to carry out our experiments: the first consists of 100,000 ratings for 1682 movies by 943 users; the second has about one million ratings for 3900 movies by 6040 users (http://www.grouplens.org). In the two datasets, each user has rated at least 20 movies, and each user rating (on a scale of 1 to 5) for an item is associated with a rating time.

Each of the two datasets is randomly divided into a training item set and a test (active) item set, with the split being defined by $Ratio = |TestItems|/|TotalItems|$. In testing each active item, we in turn assume knowledge of its first 5, 10, and 20 user ratings (used for identifying similar items), and the corresponding experiment results are denoted as Given5, Given10, and Given20, respectively. Also, we employ the standard k-means clustering algorithm. Due to space limitation, only results obtained with the first dataset are reported below. Results on the other one show a similar trend.

Measure: We use the *Mean Absolute Error* (MAE) metric to measure the prediction accuracy of our proposed algorithm. Our MAE metric is defined as

$$MAE = \frac{1}{|S|} \sum_{X \in S} |\hat{H}(X) - H(X)| \tag{8}$$

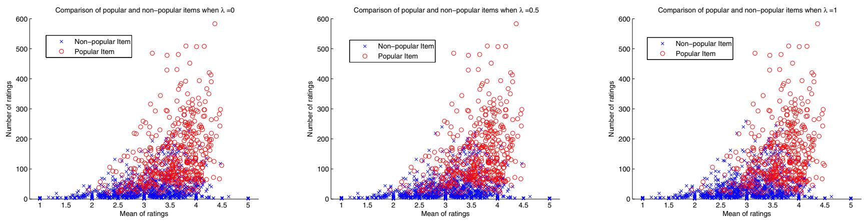
where $\hat{H}(X)$ and $H(X)$ denote the *predicted popularity* and *actual popularity* of active item X in test set S. A smaller MAE in value means a better prediction accuracy.

4.1 Experimental Results Using MAE Metric

Characteristics of Smoothing Factor λ. For the MovieLens 100,000 dataset, if a movie has received many ratings and its mean rating is high, then it is deemed to be popular. For example, some movies have been rated by more than 500 out of 943 users, and have a mean rating of 4 on a scale of 1 to 5. As explained previously, we measure the popularity of an item through the mean rating and length of its segments. δ, the significance level of the difference in mean ratings of successive segments, controls the segmentation of the rating sequence of each item. λ is a parameter for tuning the popularity measure between prolonged above-average ratings and sharp rating spikes.

We begin with several experiments to profile the impact of various λ levels on the popular items identified. We set δ =1, *# of clusters*=3, and arbitrarily fix the threshold at 0.006 (thus items with a popularity that is larger than or equal to 0.006 are identified as popular items). The results are presented in Figure 3, in which the dot and circle represent unpopular item and popular item respectively, while the X and Y axes correspond to the mean rating and number of ratings. The numbers of popular items identified are 455, 441, and 429 out of the 1682 items in the dataset, when λ is set to 0, 0.5, and 1 respectively. Although the numbers of popular items are close across different λ values, we observe that those items with longer high rating segments are more likely to be identified as popular ones with λ=0 in Figure 3(a), whereas items containing segments with high mean ratings are favored with λ=1 in Figure 3(c). When λ is set to 0.5, Figure 3(b) shows that the selection of popular items reflects a tradeoff between the mean rating and the length of the item's high-rating segments.

Characteristics of Top-K INFU. Our procedure in Section 4.2.1 usually generates a very large *CINFU* set, and it is very inefficient to use all the *CINFUs* for predicting the item popularity. So we employ a scoring method to find the top-K *CINFU* in each



(a) $\lambda = 0$ (b) $\lambda = 0.5$ (c) $\lambda = 1$

Fig. 3. Comparison between popular and non-popular items under varied λ

cluster. The underlying assumption is that if a *CINFU* u has rated many items and these items subsequently become popular, then u is likely to be a real *INFU*. The next set of experiments is designed to validate this assumption. The results in Figure 4 are obtained by measuring how many items each of the 943 users has rated (on the X axis)and the mean popularity of these items (on the Y axis), then using Equation 2 and 3 to select the 100 highest-scoring users. The parameter settings are: $ratio$=0.2, *# of clusters*=3, and $Given10$. The circles in the figure represent the top-K *INFUs*. We observe that

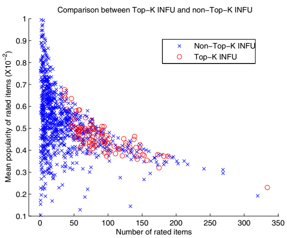


Fig. 4. Top-K INFU vs. non-Top-K INFU

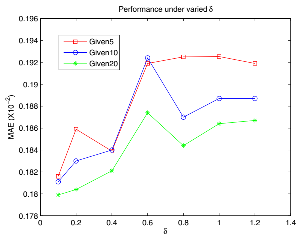


Fig. 5. Performance under different δ

many users have rated several dozens of items, while only a few users have rated more than 100 items. Each of the *INFUs* has rated relatively more items, and these items have larger mean popularities. Selection of the top-K *INFUs* in our algorithm reveals the characteristics of influential users in real life, who generally have higher impacts on certain products and are keen to comment on as many products as they can.

Impacts of δ and θ. In segmenting the rating sequences and in generating the candidate influential users, δ and θ are parameters that set the target difference in mean ratings. Several experiments are performed here in order to show the impacts of δ and θ on the prediction accuracy of our algorithm. The other parameters are fixed at $ratio$=0.2, *# of clusters*=3, λ=0.5, and Top-K users=8. The results are presented in Figure 5 which varies δ with θ=0.4, and Figure 6 which varies θ with δ=0.8.

As illustrated in Figures 5 and 6, $Given5$ results in a larger MAE, whereas $Given20$ gives the smallest MAE for the prediction results. This shows that the algorithm performs better when more user rating information is provided, which is not surprising.

As δ and θ increase, the MAE rises initially, but stabilizes after a while. This happens at $\delta > 0.6$ and $\theta > 0.6$ in Figure 5 and 6 respectively. A large δ value causes most segments with low to moderate mean rating differences to be merged, leaving only adjacent segments that have large gaps between their mean ratings. This adversely affects the identification of popular items, and translates to a fall in prediction accuracy. A similar behavior is observed for θ in Figure 6.

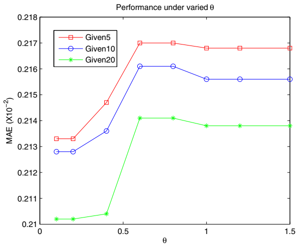


Fig. 6. Performance under different θ

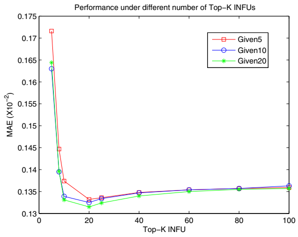


Fig. 7. Various number of Top-K INFUs

Performance under Varied Number of Top-K INFUs. When using the Top-K *INFUs* to predict the active item's popularity, we are confronted with the problem of how many *INFUs* is enough. To study this problem, we run several experiments with $ratio$=0.2, *# of clusters*=3, δ=0.8, θ=0.4, and λ=0.5. The results in Figure 4 show that for the MovieLens 100,000 dataset, the MAE drops rapidly with the initial increase in the number of Top-K *INFUs*. However, after K grows beyond 10, the MAE remains nearly unchanged, meaning that any further increase of *INFUs* does not enhance our algorithm's prediction quality.

5 Conclusions

In this paper, we proposed a novel framework for predicting popular items from historical user-item rating dataset through the help of influential users. We formulate the concepts of popular item and influential users, and quantified them with a method that is built upon the piecewise linear representation algorithm and the t-test. We then harnessed the popularity trends of similar items and predicted ratings of influential users, to predict the popularity of the target item. We have conducted extensive experiments to test the effectiveness of our framework. As an interesting enrichment for recommender systems, our framework is useful in real applications, such as web-based marketing, advertising, and personalized recommendation.

Acknowledgment. This work is supported in part by the Australian Research Council (ARC) under large grant DP0985456; the China "1000-Plan" National Distinguished Professorship; the China 863 Program under grant 2012AA011005; the China 973 Program under grants 2013CB329404, 2012CB326403; the Natural Science Foundation of China under grants 61170131, 61363009, 61263035; the Guangxi Natural Science Foundation under grant 2012GXNSFGA060004; and the Guangxi "Bagui" Teams for Innovation and Research.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-art and possible extensions. IEEE Trans. on Knowledge and Data Engineering 17(6), 734–749 (2005)
2. Archak, N., Ghose, A., Ipeirotis, P.G.: Show me the money! deriving the pricing power of product features by mining consumer review. In: Proceedings of the 13th ACM SIGKDD, pp. 56–65. ACM (August 2007)
3. Dellarocas, C., Awad, N.F., Zhang, X.: Exploring the value of online reviews to organizations: Implications for revenue forecasting and planning. Management Science, 1407–1424 (2003)
4. Doane, D.P., Seward, L.E.: Applied Statistics in Bussiness and Economics. McGraw-Hill, USA (2007)
5. Hofmann, T.: Laten semantic models for collaborative filtering. ACM Transactions on Information Systems (TOIS) 22(1), 89–115 (2004)
6. Hu, M., Liu, B.: Mining and summarizing customer reviews. In: Proceedings of the 10th ACM SIGKDD, pp. 168–177. ACM (2004)
7. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender systems: an introduction. Cambridge University Press (2010)
8. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), pp. 289–296. IEEE (2001)
9. Kohrs, A., Merialdo, B.: Clustering for collaborative filtering applications. In: Proceedings of the CIMCA (January 1999)
10. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: Proceedings of the fourth ACM International Conference on Web Search and Data Mining, pp. 287–296. ACM (2011)
11. Pu, P., Chen, L., Hu, R.: A user-centric evaluation framework for recommender systems. In: Proc. of the 5th ACM Conference on Recommender Systems, pp. 157–164. ACM (2011)
12. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of Int. Conf. of World Wide Web, pp. 285–295. ACM (2001)
13. Vany, A.D.: Hollywood economics: how extreme uncertainty shapes the film industry. Routledge, USA (2004)
14. Xue, G.R., Lin, C., Yang, Q., Xi, W., Zeng, H.J., Yu, Y., Chen, Z.: Scalable collaborative filtering using cluster-based smoothing. In: Proceedings of the 28th ACM SIGIR 2005, pp. 114–121. ACM (2005)
15. Zhang, Y., Koren, J.: Efficient bayesian hierarchical user modeling for recommendation systems. In: Proceedings of the 30th ACM SIGIR, pp. 47–54. ACM (2007)
16. Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J.R., Zhang, Y.-C.: Solving the apparent diversity-accuracy dilemma of recommender systems. PNAS 107, 4511–4515 (2010)