Enhancing Recommender Systems Under Volatile User Interest Drifts

Huanhuan Cao[1], Enhong Chen[1], Jie Yang[1], Hui Xiong[2]
[1] School of Computer Science and Technology, University of Science and Technology of China (USTC)
[2] MSIS Department, Rutgers University
E-mail: [1]{caohuan, cheneh, yangjie1}@ustc.edu.cn, [2]hxiong@rutgers.edu

ABSTRACT

This paper presents a systematic study of how to enhance recommender systems under volatile user interest drifts. A key development challenge along this line is how to track user interests dynamically. To this end, we first define four types of interest patterns to understand users' rating behaviors and analyze the properties of these patterns. We also propose a *rating graph* and *rating chain* based approach for detecting these interest patterns. For each users' rating series, a rating graph and a rating chain are constructed based on the similarities between rated items. The type of a given user's interest pattern is identified through the density of the corresponding rating graph and the continuity of the corresponding rating chain. In addition, we propose a general algorithm framework for improving recommender systems by exploiting these identified patterns. Finally, experimental results on a real-world data set show that the proposed rating graph based approach is effective for detecting user interest patterns, which in turn help to improve the performance of recommender systems.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Filtering; H.3.5 [**On-line Information Services**]: Web-based Services

General Terms

Experimentation, Algorithms

Keywords

Recommender system, interest pattern, interest drift

1. INTRODUCTION

Recommender systems [2, 18] identify user interests and provide personalized suggestions from overloaded information by exploiting the opinions of a community of users. Recent years, recommender systems have been widely used in many business applications [4, 12, 14]. In general, there are three ways to develop recommender systems. The first one is content-based (CB) [24] and targets on suggesting items which are similar to those a given user has liked in the past. The second way is based on collaborative filtering (CF) [3, 19]. In other words, recommendations are made according to the tastes of other users that are similar to the target user. Finally, a third way is to combine the above and have a hybrid solution [13].

However, traditional recommender systems usually do not consider the scenarios that users' interests drift with time. Indeed, volatile user interest drifts have been a major hinder to the successful applications of recommender systems.

In the literature, there are some emerging studies that have limited progress in detecting user interest drifts for improving recommender systems. For instance, Ding et al. [8] used a time weight method to give different weights to old and new rating data without explicitly detecting user's interest drifts. Also, a Bayesian based approach for tracking user interest drifts was proposed to improve CB recommender systems [11]. In addition, cluster-changing and auto-similarity methods have been applied for detecting user interest drifts to improve CF recommender systems [15]. However, these approaches are designed for pure CB or CF recommender systems and need complex detection modules with high computational cost. In practice, many real-world recommender systems have been developed based on hybrid methods in order to achieve better performance [2]. Therefore, it is expected to have a more flexible, and systematic method to detect user interest drifts for enhancing recommender systems. Towards this goal, there are some open questions about user interest patterns that should be first answered. Specifically, which typical patterns can summarize diverse users' ratings? How to distinguish them from each other? Whether the detection of these patterns can help improve recommender systems? If the answer is "yes", how much improvement can be achieved?

To this end, in this paper, we provide an organized study of how to improve recommender systems under volatile user interest drifts. Along this line, we first analyze time-related user rating data and introduce four types of user interest patterns to characterize the diversity of user rating behaviors. These patterns are *Single Interest Pattern (SIP)*, *Multiple Interests Pattern (MIP)*, *Interest Drift Pattern (IDP)*, and *Casual Noise Pattern* (*CNP*). In the paper, we show that both IDP and CNP have negative impacts on the performance of recommender systems. Also, to identify these patterns, we design a rating graph and rating chain based approach which includes a preprocessing stage and a detecting stage. In the preprocessing stage, given a user's corresponding rating series, the corresponding rating graph and rating chain are constructed. Then in the detecting stage, the interest patterns of the given user are detected through the analysis of the rating graph and rating chain. Finally, to improve the performance of recommender systems, we remove user

CIKM'09, November 2–6, 2009, Hong Kong, China.

ratings identified as CNP and prune user ratings identified as IDP while building recommender systems. In this way, the noisy data are removed as much as possible and the learning quality of recommend systems and the recommendation quality are both improved. To validate the proposed approach, we carry out experiments on the MovieLens data set. The results show that the precision, recall and F1-measure of user interest pattern detection are higher than 90%, and the performance of recommender systems are improved through detecting and tackling interest patterns in terms of the Hit Ratio and Macro-DOA metrics.

The contributions of this paper are summarized as follows.

We first systematically study different types of user interest patterns and propose an effective approach for detecting these patterns. We show that, among four types of patterns, IDP and CNP have negative impacts on the performance of recommender systems.

We propose a general algorithm framework based on interest drift pattern detection. This framework is suitable for most existing recommender systems.

We conduct extensive experiments to demonstrate that recommender systems can be improved through detecting and tackling the user interest patterns. The performance improvement depends on the proportion of negative interest patterns, i.e., IDP and CNP, in the raw user ratings.

Overview. The rest of this paper is organized as follows. Section 2 introduces four types of interest patterns. In Section 3, we propose an approach of detecting user interest patterns. Section 4 gives a general algorithm framework for improving existing recommender systems by leveraging interest patterns. In Section 5, we show the experimental results. Section 6 provides a brief introduction of related work. Finally, in Section 7, we conclude this paper.

2. USER INTEREST PATTERNS

We argue that for each rated item in a user rating series, there is a latent user interest that motivates the user to make the rating, and we say this item reflects the latent user interest. The lasting time of an user interest is defined as follows.

Definition 1 (Lasting time of interest). *Given a user's rating series $\mathcal{I} = I_1 I_2 ... I_N$, where $I_i (1 \leq i \leq N)$ means a rated item, the lasting time of an interest X is $(t_{end} - t_{begin})$, where t_{begin} is the timestamp of the first item reflecting X and t_{end} is the timestamp of the last item reflecting X.*

By considering lasting times of interests, we propose four patterns of user interests to summarize diverse rating series as follow:

- **Single Interest Pattern (SIP)**: Given a user's rating series $\mathcal{I} = I_1 I_2 ... I_N$, if $\forall_{1 \leq i \leq N} I_i$ reflects the same interest X, in other words, X lasts for the whole time range of $\mathcal{I}$, we say the given user's ratings satisfy the single interest pattern.
- **Multiple Interests Pattern (MIP)**: Given a user's rating series $\mathcal{I} = I_1 I_2 ... I_N$, if 1) $\exists_{1 \leq i,j \leq N}$ I_i and I_j reflect different interests; and 2) most interests reflected in $\mathcal{I}$ last for the most of $\mathcal{I}$'s time range, we say the user's ratings satisfy the multiple interests pattern.
- **Interests Drift Pattern (IDP)**: Given a user's rating series $\mathcal{I} = I_1 I_2 ... I_N$, if most interests reflected in $\mathcal{I}$ last for long enough but do not last for the most of $\mathcal{I}$'s time range, we say the user's ratings satisfy the interests drift pattern.
- **Casual Noise Pattern (CNP)**: Given a user's rating series $\mathcal{I} = I_1 I_2 ... I_N$, if most interests reflected in $\mathcal{I}$ last for very short times, we say the user's ratings satisfy the casual noise pattern.

Table 1: Some movies in MovieLens data and some main attributes of them.

ID	Movie Name	Cast	Genres
A	When Harry Met Sally	B. Crystal ...	*comedy, romance*
B	Titanic	Leonardo ...	*romance*
C	City of Angels	N. Cage ...	*romance*
D	Lolita	J. Irons ...	*drama,romance*
E	Notting Hill	J. Roberts ...	*comedy,romance*
F	Interview with Vampire	B. Pitt ...	*drama,horror*
G	Brave Heart	M. Gibson ...	*drama,war*
H	Star Gate	A. Tapping ...	*action,adventure,sci-fi*
I	E.T.	H. Thomas ...	*children,fantasy,sci-fi*
J	Star Wars V	M. Hamill ...	*action,sci-fi,war*
K	Transformers	S. LaBeouf ...	*action,sci-fi*
L	The Matrix	K. Reeves ...	*action,sci-fi*
M	Scream 2	D. Arquette ...	*horror,thriller*
N	8MM	N. Cage...	*thriller*
O	Basic Instinct	M. Douglas ...	*mystery,thriller*
P	Office Killer	C. Kane ...	*thriller*
Q	Diamonds	K. Douglas ...	*mystery*

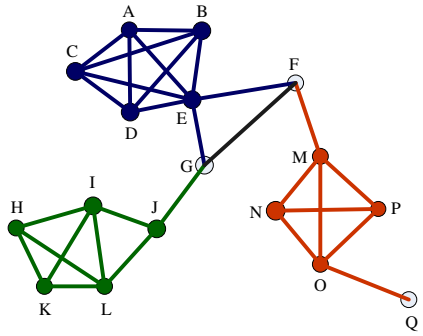


Figure 1: The graph of similarities between movies listed in Table 1.

Table 1 lists some movies from a real movies rating data set named MoveLens [1] data set. Some main attributes of these movies are shown including the movie name, the cast, the genres the movie are assigned. Figure 1 illustrates the similarity graph of these movies. In this graph, if the similarity between two movies is bigger than a predefined threshold, an edge is established between them. Similarity between two movies is evaluated according to their corresponding attributes. For instance, movie A is similar to movie B because both of them are *Romance* movies. From a similarity graph, we can see whether two movies are similar intuitively. Given a group of movies, if most of them are mutually similar, they are regarded to represent one interest. Obviously, there are three interests in the figure labeled with different colors.

Given the movies in Table 1 and their similarity graph, we give some examples of different interest patterns as follows. In these examples, we assume that the time intervals between adjacent ratings are same.

- Given a user rating series ACDEB, it's an SIP because all rated items reflect the same interest which is labeled blue in Figure 1.

- Given a user rating series AMFEN, it's an MIP because 1) this rating series reflects two interests, namely, the blue interest reflected by A, E and the red interest reflected by M, N; and 2) both interests last for the most of the user ratings' time range (4/5 time range).
- Given a user rating series AEFNM, it's an IDP because neither the blue interest reflected by A, E nor the red interest reflected by M, N last for the most of the user ratings' time range (2/5 time range).
- Given a user rating series AFOKN, it's a CNP because among the interests reflected in it, only the red interest reflected by O, N lasts for long enough (3/5 time range), but the blue interest reflected by A and the green interest reflected by K both last for only 1/5 time range.

Among these interest patterns, IDP and CNP have negative effects on the performance of recommender systems. If a user's rating series is identified as an IDP, it means the user's interests drift at some time points and the old ratings are useless since they reflect out of time interests of the user. Moreover, if a user's rating series is identified as a CNP, it means most of the user ratings are useless for characterizing the user's interests since his (or her) interests drift so frequently. For a CF-based approach, IDPs and CNPs bring noisy data in the training step and the recommendation step. Therefore, IDPs and CNPs do not only impact the quality of learning but also the recommendation accuracy. For a CB-based approach, IDPs and CNPs bring noisy data in the recommendation step while using the user ratings to represent a given user's preference model, and then impact the performance of recommender systems. For the similar reason, a hybrid approach also suffers the existences of IDPs and CNPs. In order to deal with IDPs and CNPs, we should firstly propose an effective approach for identifying different interest patterns given users' rating series. Then we should design specific methods for tackling IDPs and CNPs when building recommender systems. In the following two sections, we present our solutions for these two tasks, respectively.

3. IDENTIFYING INTEREST PATTERNS

Since the interest patterns are distinguished in terms of some subjective metrics such as "most of the user ratings' time range", it is not a simple problem to automatically detect different types of interest patterns. Therefore, we need to exploit some properties of them which can help to distinguish them and can be captured easily.

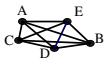
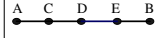
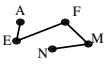
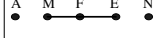
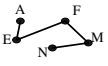
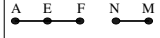
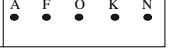
3.1 Rating Graph and Rating Chain

Before introducing the properties for distinguishing different interest patterns, we define some related notions as follows:

Definition 2 (Rating Graph Density). *Given a user's rating series $\mathcal{I} = I_1 I_2 ... I_N$, the corresponding rating graph is $G_{\mathcal{I}} = (V_{\mathcal{I}}, E_{\mathcal{I}})$, where $V_{\mathcal{I}}$ is a node set containing $I_i (1 \le i \le N)$, and $E_{\mathcal{I}} = \{e_{\mathcal{I}}\}$ is an edge set, where an edge between I_i and I_j is established if I_i and I_j are similar. The density of $G_{\mathcal{I}}$ is defined as $\frac{|E_{\mathcal{I}}|}{max_E}$, where $max_E = \binom{N}{2}$ means the max number of possible edges in $G_{\mathcal{I}}$.*

Definition 3 (Rating Chain Continuity). *Given a user's rating series $\mathcal{I} = I_1 I_2 ... I_N$, the corresponding rating chain is $C_{\mathcal{I}} = (V_{\mathcal{I}}, L_{\mathcal{I}})$, where $V_{\mathcal{I}}$ is a node set containing $I_i (1 \le i \le N)$, and $L_{\mathcal{I}} = \{l_{\mathcal{I}}\}$ is a link set, where a link can only be established between I_i and $I_{i+1} (1 \le i < N)$. If I_i and I_{i+1} are similar, a link between them is established. The continuity of $C_{\mathcal{I}}$ is defined as $\frac{|L_{\mathcal{I}}|}{N-1}$, where $N-1$ means the max number of possible links in $C_{\mathcal{I}}$.*

Table 2: Examples of interest patterns.

Pattern	Rating series	Rating graph	Rating chain
SIP	A C D E B		
MIP	A M F E N		
IDP	A E F N M		
CNP	A F O K N		

The rating graph and the rating chain are both specific cases of the similarity graph mentioned in Section 2. Different from similarity graph, they are proposed for capturing the interest coherence of rated items in a certain rating series but not all rated items. Table 2 shows the corresponding rating graphs and rating chains of the example interest patterns mentioned in Section 2.

For constructing the rating graph and rating chain of a given rating series, we need to 1) choose a similarity measure for items and 2) choose a similarity threshold to determine whether any two items are similar given the similarity measure.

As mentioned above, the similarity of two items can be measured by considering the similarities between their attributes. Without the loss of generality, the similarity between two items I_i and I_j can be defined as follows:

$$Sim(I_i, I_j) = \sum_{l=1}^{L} w_l \times Sim(A_{i,l}, A_{j,l}), \quad (1)$$

where w_l is the weight assigned to the $l-th$ attribute and $Sim(A_{i,l}, A_{j,l})$ represents the similarity between I_i and I_j on the $l-th$ attribute. For different types of attributes, the methods of calculating similarity are different. For example, for a text attribute, such as cast, genre, and key words of a movie, the similarity is calculated through the ratio of shared words between two items. By contrast, for a numeric attribute, such as the publishing year of a book, the similarity is calculated through the normalized difference of two values.

We use a method of learning the weights of similarities on each attribute proposed in Ref. [6]. Firstly, we build a training data set by selecting the items rated by enough users. Denoting the selected item set as $\mathcal{I}_{train}$, for any $I_i \in \mathcal{I}_{train}$ and $I_j \in \mathcal{I}_{train} (i \ne j)$ we establish the following equation:

$$w_1 Sim(A_{i,1}, A_{j,1}) + ... + w_l Sim(A_{i,l}, A_{j,l}) = Sim^*(I_i, I_j), \quad (2)$$

where $Sim^*(I_i, I_j)$ is the approximate similarity between I_i and I_j.

In our method, $Sim^*(I_i, I_j)$ is calculated by considering the ratio of co-ratings. Particularly, we calculate $Sim^*(I_i, I_j)$ as follows:

$$Sim^*(I_i, I_j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|},$$

where U_i means the users who rated item I_i and U_j means the users who rated item I_j. Some previous studies [19, 7] have demonstrated that two items are similar if they are usually co-rated by users. In our method, the items in $\mathcal{I}_{train}$ are all rated by many times, which further improves the confidence of the co-rating measure.

One may argue why we do not directly use $Sim^*(I_i, I_j)$ as the similarity between I_i and I_j. The problem of this method is $Sim^*(I_i, I_j)$ only makes sense if both I_i and I_j have been rated by enough times.

Otherwise, $Sim^*(I_i, I_j)$ might not be a good approximation of the real similarity between I_i and I_j, because the sparseness of I_i and I_j results in that the value of $Sim^*(I_i, I_j)$ may be largely impacted by noisy data.

Given a series of equations as Equation 2, we conduct a learning task for learning w_l through the linear regression approach. Next, we choose a similarity threshold as follows. Firstly we randomly select some item pairs and some human volunteers are required for labeling "Similar" or "Not similar" for these item pairs. Then, for each labeled item pair we calculate their similarity by the learnt similarity measure function. Finally, a threshold is determined by making the maximum agreement between human labels and the automatic similarity measure.

3.2 Identifying SIP and CNP

We can identify SIPs and CNPs directly through their rating graph densities and their rating chain continuities. For a SIP, the corresponding rating graph must have high density because all rated items in a SIP reflect the same interest. As mentioned above, if a group of movies reflect the same interest, most of them must be mutually similar, and the number of edges in the corresponding rating graph must be close to the maximum possible number. It explains why the density of a SIP's rating graph is high. By contrast, for a CNP, the corresponding rating graph must have low density because its each interest only lasts for a very short time and the rated items are divided into many interests. Consequently, there are usually few similar item pairs in a CNP, which implies that the number of edges of the corresponding rating graph is small and the corresponding density is low. For the similar reason, a SIP's rating chain must have high continuity and a CNP's rating chain must have low continuity. Table 3 shows the rating graph densities and rating chain continuities for each rating series in Table 2. We can clearly see that for a SIP ACDEB, its rating graph density and rating chain continuity are both very high, and for a CNP AFOKN, both its rating graph density and rating chain continuity are very low. Notice that the ranges of rating graph density and rating chain continuity are both [0,1].

Table 3: The rating graph densities and rating chain continuities of the rating series in Table 2

Rating series	Rating graph density	Rating chain continuity
ACDEB	$10/\binom{5}{2} = 1.0$	$4/(5-1) = 1.0$
AMFEN	$4/\binom{5}{2} = 0.4$	$2/(5-1) = 0.5$
AEFNM	$4/\binom{5}{2} = 0.4$	$3/(5-1) = 0.75$
AFOKN	$1/\binom{5}{2} = 0.1$	$0/(5-1) = 0$

However, it is still a problem of determining appropriate thresholds of rating graph density and rating chain continuity for identifying CNPs and SIPs. For this concern, we firstly make the following assumptions: 1) In a CNP's rating graph, the probability that there exists an edge between two given items is independent and identically distributed (i.i.d.). We denote this probability as P_{CNP}; 2) In a SIP's rating graph, the probability that there exists an edge between two given items is independent and identically distributed (i.i.d.). We denote this probability as P_{SIP}.

If we randomly rate N items, which corresponds to an ideal CNP, the number of edges of the corresponding rating graph is $\binom{N}{2} \times P_{CNP}$, following the above assumptions. According to Definition 2, the density of the rating graph is:

$$Density_{CNP} = \frac{\binom{N}{2} \times P_{CNP}}{\binom{N}{2}} = P_{CNP}.$$

Similarly, according to Definition 3 the continuity of an ideal CNP's rating chain is:

$$Continuity_{CNP} = \frac{(N-1) \times P_{CNP}}{(N-1)} = P_{CNP}.$$

Moreover, if we rate N items with the same interest, which corresponds to a SIP, it's easy to derive that in the ideal scenario the density of the rating graph is P_{SIP}, and the continuity of rating chain is P_{SIP} as well. However, we should consider noise because occasionally a user may carelessly rate items which look like in line with his (or her) interest but are actually not. Denote a user defined noisy factor as α, the density of an ideal SIP's rating graph is as follow:

$$Density_{SIP} = \frac{P_{SIP}\binom{(1-\alpha)\cdot N}{2} + P_{CNP}\binom{\alpha \cdot N}{2} + P_{CNP}(1-\alpha)\alpha \cdot N^2}{\binom{N}{2}},$$

where $\alpha \cdot N$ means the number of randomly rated items, and $(1-\alpha) \cdot N$ means the number of rated items that reflect the same interest.

Similarly, the continuity of an ideal SIP rating chain is:

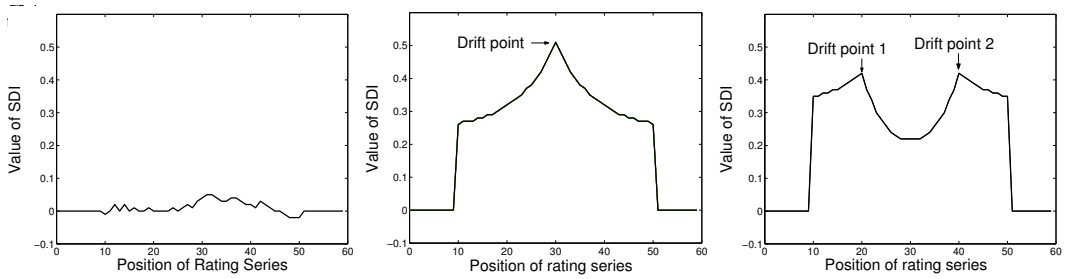
$$\begin{aligned} Continuity_{SIP} &= \frac{(N-1) \times ((1-\alpha) \cdot P_{SIP} + \alpha \cdot P_{CNP})}{(N-1)} \\ &= (1-\alpha) \cdot P_{SIP} + \alpha \cdot P_{CNP} \end{aligned}$$

Therefore, the thresholds of rating graph density and rating chain continuity for distinguishing SIPs and CNPs from other interest patterns can be calculated through P_{CNP} and P_{SIP}. Given a rating series, if its rating graph density is smaller than $Density_{CNP}$ and its rating chain continuity is smaller than $Continuity_{CNP}$, it is identified as a CNP. Otherwise, if its rating graph density is bigger than $Density_{SIP}$ and its rating chain continuity is bigger than $Continuity_{SIP}$, it is identified as a SIP. We consider both of the rating graph's $Density_{SIP}$ and the rating chain's $Continuity_{SIP}$ of a rating series because both of them are approximate features of CNP and ISP. Combining them may enhance the precision of detecting CNP and ISP.

The values of P_{CNP} and P_{SIP} can be statistically estimated for a specific data set. For example, in our experiments on the MovieLens data, we randomly select 5,000 pairs of items and calculate the ratio that the corresponding similarity is bigger than a predefined threshold, where an edge can be established. The same sampling process is repeated ten times and we estimate P_{CNP} through the mean ratio. For P_{SIP}, we take advantage of the item pairs labeled "Similar" by volunteers, which is mentioned in Section 3.1. We randomly sample 500 item pairs from the item pairs labeled "Similar" by ten times, and estimate P_{SIP} through the mean ratio that the similarity between a pair of items is bigger than the predefined threshold of similarity.

3.3 Identifying MIP and IDP

Though SIP and CNP can be directly identified through their rating graph densities and rating chain continuities, there is still a challenge to distinguish the last two interest patterns, i.e., MIP and IDP. Before presenting our algorithm for distinguishing them, let us look into an property of IDP. In an IDP, few interests last for the most of the whole time range. In other words, in an IDP, some interests disappear and some interests begin to appear as time goes on, which implies that there should be some interests transaction



(a) No interest drift. (b) One interest drift. (c) Two interest drifts.

Figure 2: The SDI series of three synthetic interest patterns.

zones in the rating series where both old and new interests appear. If we split an IDP at some position in one of its interests transaction zones, the split segments' rating graphs usually are more dense than the one of original rating series, because their ratings are more coherent in terms of interests. By contrast, MIP does not have this property since most of its interests last for the most of the whole time range. Based on this property, we propose a heuristic segmentation method for splitting the rating series at the positions where a split will cause large increase of rating graph density.

We use *Splitting Density Increment(SDI)* to measure the increase of rating graph density after splitting. Given a rating series $\mathcal{I}$, if we split $\mathcal{I}$ at the i-th position, we denote the left sub-series as $\mathcal{I}i\rfloor$ and the right sub-series as $\mathcal{I}\lfloor i$, then the SDI at the i-th position is calculated as follows:

$$SDI_i = \frac{Density_{\mathcal{I}i\rfloor} + Density_{\mathcal{I}\lfloor i}}{2} - Density_{\mathcal{I}}, \tag{3}$$

where $Density_{\mathcal{I}i\rfloor}$ means the density of $\mathcal{I}i\rfloor$'s rating graph, $Density_{\mathcal{I}\lfloor i}$ means the density of the $\mathcal{I}\lfloor i$'s rating graph, and $Density_{\mathcal{I}}$ means the density of $\mathcal{I}$'s rating graph, respectively.

The main steps of our algorithm are listed as follows. Firstly, we find the position with the max SDI. If it is a *drift point*, the rating series is split at this position. Then the same procedure is recursively performed for split rating sub-series. Otherwise, if the position with max SDI is not a drift point, the algorithm terminates. A drift point implies that the user's interest probably drifts at this position. After executing the algorithm, if no drift point can be found, we regard the given rating series as a MIP. Otherwise, we regard it as an IDP and the drift points are used for further processing as mentioned in Section 4.

We determine whether the position with max SDI is a drift point by referring an approach proposed in Ref. [17]. The main idea of this approach is as follow. Given a series S, for the i-th position, we define:

$$t_i = F(\mu_{Si\rfloor}, \mu_{S\lfloor i}),$$

where μ_* is a feature of $*$ and $F(a, b)$ is a bivariate function. Then a t series can be generated. Denoting the maximum value in the t series as t_{max}, we define the probability that t_{max} is bigger than the maximum value of a random series with the same length as $P(t_{max})$. If $P(t_{max})$ is bigger than a predefined significance level P_0, the corresponding position of t_{max} is regarded as a peak. In practice, P_0 is usually set to be 0.95. Given a numeric series with N elements, $P(t_{max})$ can be calculated as follow:

$$P(t_{max}) = [1 - B_{v/(v+t^2_{max})}(\delta v, \delta)]^{\gamma}, \tag{4}$$

where $B_x(a, b)$ is the incomplete beta function, $v = N - 2$, $\gamma = \sigma lnN - \beta$, and σ, β, δ are constant values and usually approximated by Monte Carlo simulations.

This approach is widely used for determining whether or not segmenting a series by the existence of a peak of t. In our application, the SDI corresponds to t. Given a rating series, the corresponding SDI series can be generated. Denote the position with SDI_{max} as p_{max}, if $P(SDI_{max})$ is bigger than P_0, we can state that p_{max} is a peak, which implies the corresponding increase of rating graph density is significant and p_{max} can be regarded as a drift point.

Examples in Figure 2 may help to understand the notions of SDI and drift points. These figures show the SDI series of some synthetic interest patterns. The method of generating synthetic interest patterns is introduced in Section 5.3. Figure 2 (a) shows the SDI series of a syntectic MIP with 60 rated items and 2 interests. We can see this SDI series has no obvious peak. Figure 2 (b) shows the SDI series a synthetic SIP with one interest drift. We can see there is a peak in this SDI series, which is a drift point. Figure 2 (c) shows the SDI series of a synthetic SIP with two interest drifts. We can see there are two peaks in this SDI series, which are both drift points.

When splitting the rating series, we need to constrain the minimum length of a sub-series because a too short rating series may not reflect the user's interests well. Such a constraint depends on specific applications. For example, GroupLens research group states that it at least needs 20 rated movies to reflect the interests of a user [19].

Algorithm 1 shows the details of the Density Based Segmentation(DBS) method, where $\mathcal{I}$ is an item series sorted by rating time for a user, p_{begin} is a subseries's first position in $\mathcal{I}$, and p_{end} is a subseries's last position in $\mathcal{I}$. Moreover, $\mathcal{I}' = \mathcal{I}[p_{begin}, p_{end}]$ means $\mathcal{I}$'s subseries from p_{begin} to p_{end}, and L_0 means the predefined minimum length of a split sub-series. The sub-method $SDI(\mathcal{I}_1, \mathcal{I}_2)$ computes the SDI of two neighboring series $\mathcal{I}_1$ and $\mathcal{I}_2$.

Time Complexity Analysis. Given a rating series $\mathcal{I}$ with N rated items, when calculating the SDI of each position i of $\mathcal{I}$, we need to calculate $Density_{\mathcal{I}}$, $Density_{\mathcal{I}i\rfloor}$, and $Density_{\mathcal{I}\lfloor i}$, respectively. In a naive algorithm, the calculations of $Density_{\mathcal{I}i\rfloor}$ and $Density_{\mathcal{I}\lfloor i}$ need $O(i^2)$ and $O((N-i)^2)$ time complexities, respectively. However, if the edge numbers of $\mathcal{I}i\rfloor$ and $\mathcal{I}\lfloor i$'s rating graphs are remembered, the calculations of $Density_{\mathcal{I}i\rfloor}$ and $Density_{\mathcal{I}\lfloor i}$ only need $O(i)$ and $O(N-i)$ time complexities, respectively. Because they can be computed as follow:

$$Density_{\mathcal{I}i\rfloor} = \frac{E_{\mathcal{I}(i-1)\rfloor} + E^+_{\mathcal{I}i}}{\binom{i-1}{2}}, \tag{5}$$

$$Density_{\mathcal{I}\lfloor i} = \frac{E_{\mathcal{I}\lfloor(i-1)} + E^-_{\mathcal{I}i}}{\binom{N-i+1}{2}}, \tag{6}$$

where $E_{\mathcal{I}(i-1)\rfloor}$ means the edge number of $\mathcal{I}(i-1)\rfloor$'s rating graph,

Algorithm 1: $DBS(\mathcal{I}, p_{begin}, p_{end})$

input : $\mathcal{I}$ is the original item series for a given user;
p_{begin} is subseries's first position in $\mathcal{I}$;
p_{end} is subseries's last position in $\mathcal{I}$;
output: the set of drift positions Ψ;

```
1  Initialization: Ψ = Φ; I' = I[p_begin, p_end];
2  if (p_begin − p_end) < 2 × L_0 then
3      return Ψ;
4  i = p_begin + L_0;
5  while i ⩽ p_end − L_0 do
6      SDI_i = SDI(I'⌊i⌋, I'⌊i⌋);
7      i = i + 1;
8  SDI_max = the max of SDI_i, i ∈ [p_begine + L_0, p_end − L_0];
9  p_max = the position with SDI_max;
10 if P(SDI_max) ⩾ P_0 then
11     Ψ∪ = {p_max};
12     Ψ∪ =DBS(I, p_begin, p_max);
13     Ψ∪ =DBS(I, p_max, p_end);
14 return Ψ;
```

$E_{\mathcal{I}\lfloor(i-1)}$ means the edge number of $\mathcal{I}\lfloor(i-1)$'s rating graph, $E^+_{\mathcal{I}i}$ means the number of edges that connect the i-th item with it's previous items, and $E^-_{\mathcal{I}i}$ means the number of edges that connect the i-th item with it's following items. Obviously, the computations of $E^+_{\mathcal{I}i}$ and $E^-_{\mathcal{I}i}$ need to scan i and $N-i$ items. Thus, the time complexities of computing Equation 5 and Equation 6 are $O(i)$ and $O(N-i)$, respectively. Furthermore, the whole scan of $\mathcal{I}$ computes SDIs N times, so the corresponding time complexity is $O(N \times O(i) + N \times O(N-i)) = O(N^2)$. In the worst case, the original rating series can be split into single items, and the corresponding time complexity is $O(N^2 logN)$. However, since we have the threshold of minimum length of rating sub-series and a split happens only when there is a drift point, the practical time cost is much less than that of the worst case.

4. IMPROVE RECOMMENDER SYSTEMS

We propose a general algorithm framework to improve the performance of recommender systems by detecting different types of interest patterns and invoking the corresponding operations. This framework consists of an off-line part and an on line part, as illustrated in Figure 3.

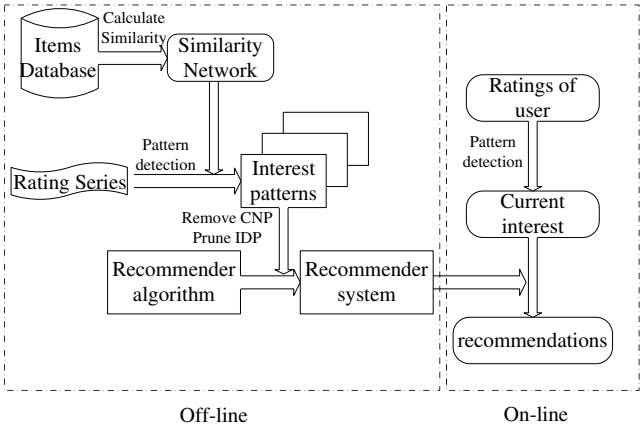


Figure 3: The framework of applying user interest pattern detection to improve recommender systems.

The off-line part has two steps, namely, the preprocessing step and the recommender constructing step. In the preprocessing step, we prune the users' rating series for reducing noisy data. Firstly, we extract the attributes of items and calculate the similarities between items. A $N \times N$ similarity matrix $SimM$ is initialized as $\forall_{i,j} SimM[i][j] = 0$. If the similarity between I_i and I_j is larger than a predefined threshold, we set $SimM[i][j] = 1$. Then we identify different user interest patterns from all users' rating series. If a user's rating series is identified as a CNP, it will be removed. Otherwise, if it is identified as an IDP, we split it at the most recent drift point P_{recent} and drop the sub-series in front of P_{recent}. It is because these sub-series are regarded as out of time for reflecting the user's interests. For SIPs and MIPs, we do nothing with them since they have no negative effect on recommender systems. In the recommender constructing step, we construct a recommender system based on an existing recommender algorithm from the pruned rating series. Any CF or Hybrid based recommender systems can benefit from the preprocessing step because the reduction of noisy data improves the quality of training data, so does the quality of learning.

In the on-line part, the framework makes recommendations for users by considering their rating series. Given a user, we firstly detect his (or her) interest pattern through the corresponding rating series. In the first case, if the user's rating series is a SIP or MIP we make recommendations by considering whole of the corresponding rating series. In the second case, if the user's rating series is a CNP we make recommendations without considering the corresponding rating series and just recommend the most popular items. In the final case, if the user's rating series is an IDP, we make recommendations only considering the rating sub-series behind the most recent drift point. Any CF, CB, or Hybrid based recommender systems can benefit from this part because the noisy data are removed from the user's rating series.

The major advantage of this framework is that it can be used for improving a wide range of existing recommender systems. It is like a "wrapper" which contains an existing recommender system as a module. Moreover, an additional interest pattern detection & operation module is integrated to improve the performance of the original recommender system. Our experiments show that such improvements are usually significant.

5. EXPERIMENTAL RESULTS

In this section, we report the experimental results on a real data set and several semi-synthetic data sets, which show that our approach for detecting interest patterns is effective and the proposed interest pattern based algorithm framework can significantly improve a wide range of recommender systems.

5.1 Experimental Data

We conduct extensive experiments on the MovieLens [1] data set, which is a widely used benchmark data set for evaluating the performance of recommender systems. The MovieLens data set includes over 1,000,000 ratings from 6,040 users for 3,900 movies. Each movie is associated with the ID, the movie name, the release year, the script writers, the directors and the genres. There are 18 genres for all the movies, and each movie belongs to one or more genres. The rating data contain one million ratings and each rating is associated with the user ID, the movie ID, the rating value, and the timestamp.

5.2 Parameter Selection

Our approach has several parameters for detecting interest patterns. In this section, we report how to determine these parameters for the MovieLens Data.

5.2.1 Similarity

Constructions of rating graphs and rating chains are based on a similarity measure and the corresponding similarity threshold. As mentioned in Section 3.1, we use a linear similarity function on each attribute as the similarity measure of items. The weight of each attribute is learnt through the linear regression approach. In our experiments, the learnt weight for each attribute is shown in Table 4. From this table we can see, genres and cast are the most important attributes for distinguishing movies, which is consistent with common sense. Comparatively, the attribute of key words seem less important. It may be because in the MovieLens data, the key words of a movie are tagged by different users and there is no unified standard of tagging. As a result, two similar movies may be tagged with different key words by different users.

Table 4: Weight of each attribute.

Attribute	Weight
Cast	0.477
Genres	0.354
Directors	0.075
Key words	0.041
Release time	0.012
Script writers	0.011
Language	-0.017

Then we randomly select 5,000 movie pairs and require three college volunteers to label whether the two movies in a movie pair are similar. These volunteers are all movie fans and can use search engine to retrieval the information of their unfamiliar movies. The labeled result shows 22% of movie pairs are labeled similar. The learnt similarity threshold is 0.3, which is surprisingly small at the first glance. But after careful analysis, we find that it is reasonable. Suppose two movies have the same genres but different cast, different names, and different directors. In practice, such two movies are usually considered similar while their similarity under our measure is just close to 0.35.

5.2.2 P_{CNP} and P_{SIP}

In our approach, we identify SIPs and CNPs through $Density_{SIP}$, $Density_{CNP}$, $Continuity_{SIP}$ and $Continuity_{SIP}$, respectively. According to their definitions, all of these parameters are computed based on P_{CNP} and P_{SIP}. To estimate P_{CNP}, we randomly select 5,000 pairs of items and calculate the ratio that the corresponding similarity is bigger than a predefined threshold. The same sampling process is repeated ten times and we estimate P_{CNP} through the mean ratio. To estimate P_{SIP}, we take advantage of the item pairs labeled "Similar" by volunteers. After randomly sampling 500 item pairs from the item pairs which are labeled "Similar" by ten times, the mean ratio that the similarity between a pair of items is bigger than the predefined threshold of similarity (0.3) is used as the approximation of P_{SIP}. Table 5 shows the ranges of rating graph density and rating chain continuity for each type of interest pattern.

Table 5: The ranges of rating graph density and rating chain continuity for each type of interest patterns

Pattern	Rating graph density	Rating chain continuity
SIP	$\geqslant 0.8$	$\geqslant 0.8$
MIP	$0.25 \sim 0.8$	$0.25 \sim 0.8$
IDP	$0.25 \sim 0.8$	$0.25 \sim 0.8$
CNP	$\leqslant 0.25$	$\leqslant 0.25$

5.3 Evaluation of The Approach for Detecting Interest Patterns

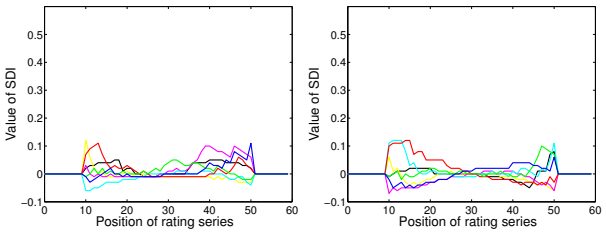
To evaluate the effectiveness of our approach for detecting interest patterns, we need a test data set which contains rating series with labels of the corresponding types of interest patterns. However, the rating series in MovieLens data have no such labels. Alternatively, we generate synthetic interest patterns based on the movies in MovieLens data. As mentioned in Section 5.2.1, both genres and cast are important attributes of movies. If several movies have the same genres, they probably reflect the same interest. Similarly, if several movies have the same cast, they also probably reflect the same interest. Since in practice it is more common that two movies have exactly the same genres than that they have exactly the same cast, we generate interest patterns by using genres to simulate interests. Particularly, to generate a SIP, we firstly randomly select a genre and then randomly select some movies with this genre to construct a rating series. To generate a MIP, we randomly select 2 or 4 genres and randomly select movies with these genres to construct a rating series. If the first or last five movies don't cover all genres in the whole rating series, we drop this rating series and repeat the above step. To generate a IDP, we firstly generate a MIP or IDP as a prefix. Then we generate another MIP or IDP with different genres and append it to the prefix. To generate a CNP, we just randomly select some movies to construct a rating series. In our experiment, each rated item is not associated with rating scores. For each type of interest patterns, we generate 100 synthetic samples with 60 movies. All generated interest patterns have been manually checked to assure the correctness of their labels.

We generate three test data sets with the above approach to evaluate our approach for detecting interest patterns. For each data set, we firstly identify CNPs and SIPs through their rating graph densities and the rating chain continuities. Then, we omit the recognized CNPs and SIPs, and use the DBS algorithm to detect IDPs. Table 6 shows the precision, recall and F1-measure of our approach on each data set. From this table we can see the precision, recall, and F1-measure for each type of interest patterns are all higher than 90%, which means our approach for detecting interest patterns is effective. Notice that even though the test data sets are generated by using genres to simulate interests, our approach is not aware of this knowledge. Instead, our approach works because the rating series' features, i.e., rating graph density and rating chain continuity, can capture the coherence of rated items on interests.

Table 6: Results of detecting interest patterns.

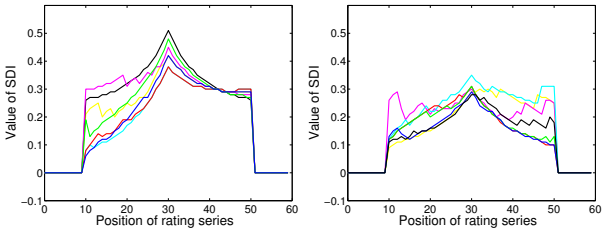
Metric	Pattern	Data set 1	Data set 2	Data set 3	Mean
Precision(%)	SIP	96.0	96.0	95.0	95.7
	CNP	93.9	93.0	93.0	93.3
	MIP	92.0	92.9	92.0	92.3
	IDP	93.1	92.1	92.9	92.7
Recall(%)	SIP	97	96	96	96.3
	CNP	92	93	93	92.7
	MIP	92	92	92	92.0
	IDP	94	93	92	93.0
F1-measure(%)	SIP	96.5	96.0	95.5	96.0
	CNP	92.9	93.0	93.0	93.0
	MIP	92.0	92.4	92.0	92.1
	IDP	93.5	92.5	92.4	92.8

Figure 4 and Figure 5 show the SDI series for some synthetic interest patterns in the experiment. In each figure, we show the SDI series of 7 synthetic rating series which are the same interest patterns. From these figures we can see IDPs have obvious peaks of SDIs while MIPs have no such peaks, which intuitively explains why DBS algorithm works.



(a) MIP with 2 interests. (b) MIP with 3 interests.

Figure 4: Without interest drift there is no peak of SDI.



(a) IDP from 2 to 1. (b) IDP from 2 to 2.

Figure 5: An interest drift exists at the 30^{th} point, and SDI reaches the maximum value there.

5.4 Improving the Recommender Systems

In this Section, we evaluate the effectiveness of enhancing recommender systems with interest pattern detection.

5.4.1 Experiment Set Up

The proportion of negative interest patterns in the original MovieLens data is fixed. To study the effect of detecting and tackling negative interest patterns on recommender systems with different proportions of negative interest patterns, we also generate some semi-synthetic data sets as follows. Firstly, we identify all SIPs, MIPs, IDPs, CNPs in the MovieLens data. Then we remove different numbers of MIPs and SIPs from the original data to build additional data sets. The MIPs and SIPs to be removed are randomly selected with the same proportion in the original data.

Given a data set, for each rating series, we extract the last 20% ratings as the test set and use the sub-series containing the first 80% ratings as the training set. Recommender systems are built based on all users' training sets and then tested for each user.

We carry out experiments for three basic *k Nearest Neighbor* (*kNN*) based recommender methods including a CB method and the other two CF methods. The CB method is the *kNN*-CB (*CB* for short) proposed in Ref. [5]. The two CF methods are item-based *kNN*-CF (*IBCF* for short) and user-based *kNN*-CF (*UBCF* for short) [16], respectively. These methods are used as baseline methods. For each baseline method, we build the corresponding Interest Pattern Detection Based (IPDB) extension which firstly detects CNPs, IDPs and invokes the corresponding operations, then builds a recommender system by the baseline method.

5.4.2 Evaluation Metrics

To evaluate the performance of recommender systems, we use a coverage metric and a ranking metric as follow:

Hit Ratio(HR) is widely used to measure the accuracy of prediction on top-K recommendations. For a user u, the HR for u is defined as $\text{HR}_u = \frac{R_u \cap T_u}{K}$, where R_u represents the recommendation set and T_u represents the test set. Moreover, the HR for N users is defined as HR= $\frac{\sum HR_u}{N}$.

Degree of Agreement(DOA) is a metric of measuring how good an item ranking is for a given user. Firstly, we denote $\mathcal{M}$ as the set of all items, denote $\mathcal{M}_u$ as the set of the items which the user u has rated, and denote $\overline{\mathcal{M}_u}$ as the set of the items which u has not rated, where $\overline{\mathcal{M}_u} = \mathcal{M} - \mathcal{M}_u$. Then we define the boolean function:

$$check\_order_u(I_j, I_k) = \begin{cases} 1 & if \quad Rank_u(I_j) \geq Rank_u(I_k) \\ 0 & otherwise \end{cases},$$

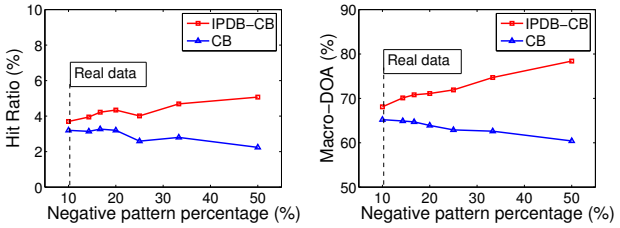
where $Rank_u(I_j)$ means the rank of item I_j in the ranked recommendation list for user u. In the end, DOA for user u can be computed by:

$$\text{DOA}_u = \frac{\sum_{I_j \in T_u, I_k \in \overline{\mathcal{M}_u}} check\_order_{u_i}(I_j, I_k)}{|T_u| \cdot |\overline{\mathcal{M}_u}|}$$

DOA_u measures for user u the percentage of item pairs ranked in the correct order with respect to the total number of pairs. A good recommender system should rank the items that have indeed been rated at higher positions than items that have not been rated. A global degree of agreement for N users, named Macro-averaged DOA (or shortly Macro-DOA), is computed as Marco-DOA = $\frac{\sum \text{DOA}_u}{|N|}$

5.4.3 Experimental Results

Firstly we compare the performance of CB and its IPDB extension (*IPDB-CB* for short). Figure 6 (a) and Figure 6 (b) show the HR and Macro-DOA of the two methods on data sets with different percentages of negative interest patterns, respectively. From these two figures we can see that the performance of CB drop slowly in terms of HR and Macro-DOA as the percentage of negative interest patterns increases, which illustrates that the negative interest patterns indeed have negative impacts on the performance of recommender systems. We can also see that IPDB-CB always outperforms CB in terms of HR and Macro-DOA. The minimum and maximum improvements in terms of HR are 15.6% and 126.3%, and the minimum and maximum improvements in terms of Macro-DOA are 4.4% and 29.8% respectively. Moreover, the improvement of IPDB-CB to CB increases with the increase of the percentage of negative interest patterns in terms of both HR and Macro-DOA.



(a) (b)

Figure 6: Compare the performance of CB and IPDB-CB in terms of HR and Macro-DOA.

Then we compare the performance of IBCF and its IPDB extension (*IPDB-IBCF* for short). Figure 7 (a) and Figure 7 (b) show the HR and Macro-DOA of the two methods on data sets with different percentages of negative interest patterns, respectively. From these two figures we can see that the performance of IPDB-IBCF and

IBCF both drop in terms of HR and Macro-DOA as the percentage of negative interest patterns increases. However, IPDB-IBCF always outperforms IBCF in terms of HR and Macro-DOA. The minimum and maximum improvements in terms of HR are 2.9% and 7.9%, and the minimum and maximum improvement in terms of Macro-DOA are 1.2% and 6.7%, respectively. Moreover, the improvement of IPDB-IBCF to IBCF increases with the increase of the percentage of negative interest patterns in terms of both HR and Macro-DOA.

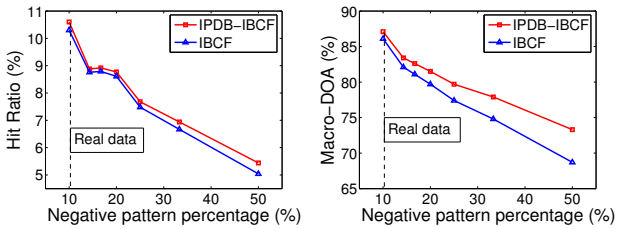


Figure 7: Compare the performance of IBCF and IPDB-IBCF in terms of HR and Macro-DOA.

Finally, we compare the performance of UBCF and its IPDB extension (*IPDB-UBCF* for short). Figure 8 (a) and Figure 8 (b) show the HR and Macro-DOA of the two methods on data sets with different percentages of negative interest patterns, respectively. From these two figures we can see that the performance of IPDB-UBCF and UBCF both drop in terms of HR and Macro-DOA as the percentage of negative interest patterns increases. However, IPDB-UBCF always outperforms UBCF in terms of HR and Macro-DOA. The minimum and maximum improvements in terms of HR are 2.0% and 9.3%, and the minimum and maximum improvement in terms of Macro-DOA are 0.3% and 2.1%, respectively. Moreover, the improvement of IPDB-UBCF to UBCF increases with the increase of the percentage of negative interest patterns in terms of both HR and Macro-DOA.

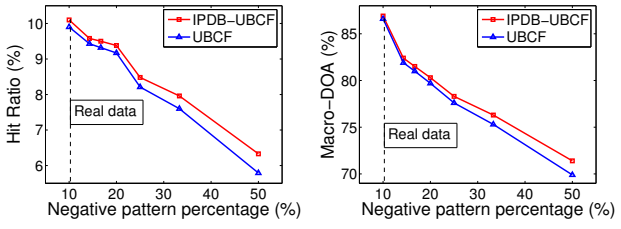


Figure 8: Compare the performance of UBCF and IPDB-UBCF in terms of HR and Macro-DOA.

Based on above experimental results, we can state that negative interest patterns have negative impacts on recommender systems and our interest pattern detection based algorithm framework is effective for improving a wide range of existing recommender algorithms, including CB-based algorithms and CF-based algorithms. Moreover, the bigger percentage of negative interest patterns is, the bigger improvement our approach can achieve. It is easy to understand because our approach is effective for dealing with negative interest patterns and can ensure stable performance with the increase of negative interest patterns. By contrast, the base lines suffer more as the percentage of negative interest patterns increases. Last, the improvement on CB-based algorithms is big than that on CF-based. It may be because CB-based algorithm only depends single user's ratings and is sensitive to negative interest patterns, and CF-based algorithms is less sensitive to negative interest patterns by considering collaborative information from similar users.

6. RELATED WORK

Related literature on recommender systems can be grouped into three categories based on how recommendations are made [2, 18]. Specifically, three categories are Content Based methods (CB) [6], Collaborative Filtering methods(CF) [3, 8, 12, 19] and Hybrid methods [13]. In the CB model, items are recommended to a user based on the preferred similar items of this user in the past. Due to limitations of textual analysis, this approach is domain sensitive, quality indistinguishable, and has the problem of overspecialization [2]. Collaborative Filtering methods can be divided into two categories, namely, user-based CF and item-based CF. In the user-based CF, items are recommended to the users based on the choices of other users who have similar tastes and preferences as this user in the past. The goal is to find similarities among users using item ratings data so that items can be recommended based on the similarities [3]. In the item-based CF, similarities between items are calculated according to the rated items by the same user. Then recommendations similar to one's rated items are produced. A hybrid method is constructed by using both of CB and CF methods.

A few researchers have studied detecting users' interest drifts in a recommender system. For example, Lam et al. [11] investigated changes of user interests in information filtering systems, and a Bayesian-based technique for tracking users' interest drifts is developed. Moreover, researchers have addressed the problem of changing user interests by decomposing an interest category into long-term and short-term interest models, relearning examples of recent window, applying decay functions, or employing evolutionary algorithms [23]. However, none of these works made a systematic study of different types of user interest patterns, where some types of interest patterns reflect user interest drifts. Indeed, in this paper, we not only propose four typical types of user interest patterns and the corresponding detection approach, but also propose a general algorithm framework that can improve a wide range of existing recommender systems by detecting user interest patterns.

Finally, recent years have witnessed increased interests in mining concept-drifting data. Concept-drifting means the concept that we try to learn from the data is constantly evolving. This is very similar to the interest-drifting in the context of this paper. For instance, there have been some efforts dedicated to data selection in the environment of concept-drifting [9, 22]. In Ref. [9], the author pointed out that using old data blindly is not better than "gambling". In other words, using old data without selectivity helps to produce a more accurate hypothesis only if there is no concept-drifting and the amount of old data arbitrarily chosen just happens to be right. In contrast, the approach proposed in this paper can select data wisely and trace changes in user interest, thus alleviating the problem of user interest drifts.

7. CONCLUDING REMARKS

In this paper, we provide an organized study of how to improve recommender systems under user volatile interest drifts. Specifically, we propose four typical types of user interest patterns which can be detected by exploiting user rating graphs and rating chains. As shown in the paper, both interest drift patterns (IDP) and casual noise patterns (CNP) have negative impacts on the performance of a wide range of recommender systems, including CBs, CFs, and

hybrid methods. Also, experimental results on a real-world data set show that, once these two types of patterns have been detected and processed with specific operations, the performance of these recommender systems can be improved as measured by Hit Ratio and Macro-DOA metrics.

8. ACKNOWLEDGEMENT

The work is supported by the National Natural Science Foundation of China (No. 60775037), the National High Technology Research and Development Program of China (863 Program) (No.2009 AA01Z123), Nokia Research Center China, and the Graduate Innovation Foundation of USTC. We also thank MovieLens group for sharing their data for research.

9. REFERENCES

[1] *http://www.grouplens.org/node/73#attachments*.

[2] Adomavicius, G., Alexander, and Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. In *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pages: 734-749, 2005.

[3] Ahn, H. J., A new similarity measure for collaborative filtering to alleviate the new user cold starting problem. In *Information Science*, vol. 178, no. 1, pages: 37-51, 2008.

[4] Billsus, D., Brunk, C. A., Evans, C., Gladish, B., and Pazzani, M., Adaptive interfaces for ubiquitous web access. In *Comunications of the ACM*, vol. 45, no. 5, pages: 34-38, 2002.

[5] Chen, J., Yin, J., and Huang, J., Automatic content-based recommendation in e-Commerce. In *Proceedings of the 2005 IEEE international Conference on e-Technology, e-Commerce and e-Service*, pages: 748-753, 2005.

[6] Debnath, S., Ganguly, N, and Mitra, P., Feature weighting in content based recommendation systems using social network analysis. In *Proceedings of the 17th international conference on World Wide Web, Beijing, China*, pages: 1041-1042, 2008.

[7] Deshpande, Mukund and Karypis, George, Item-based top-N recommendation algorithms. In *ACM Transactions on Information Systems*, vol. 22, no. 1, pages: 143-177, 2004.

[8] Ding, Yi, and Li, Xue, Time Weight Collaborative Filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management, Bremen, Germany*, pages: 485-492, 2005.

[9] Fan, W., Systematic data selection to mine concept-drifting data streams, In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, Seattle, WA, USA*, pages: 128-137, 2004.

[10] Gori, Marco, and Pucci, Augusto, A random walk based scoring algorithm with application to recommender systems for large scale E-commerce. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages: 127-146, 2006.

[11] Lam, Wai, Mostafa, and Javed, Modeling user interest shift using a bayesian approach. In *Journal of the American Society for Information Science and Technology*, vol. 52, no. 5, pages: 416-429, 2001.

[12] Linden, G., Smith, B., and York, J., Amazon.com recommendations: item-to-item collaborative filtering. In *IEEE Internet Computing*, vol. 7, no. 1, pages: 76-80, 2003.

[13] Melville, P., Mooney, R.J., and Nagarajan, R., Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the 18th National Conference on Artificial Intelligence, Edmonton, Alberta, Canada*, pages: 187-192, 2002.

[14] Miller, B. N., Albert, I., Lam, S. K., Konstan, J. A., and Riedl, J., MovieLens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th International Conference on Intelligent User Interfaces,Miami, Florida, USA*, pages: 263-266, 2003.

[15] Min, Sung-Hwan, and Han, Ingoo, Detection of the customer time-variant pattern for improving recommender systems. In *Expert Systems with Applications*, vol. 28, no. 2, pages: 189-199, 2005.

[16] Mobasher, B., Dai, H., Luo, T., and Nakagawa, M., Improving the effectiveness of collaborative filtering on anonymous web usage Data. In *Proceedings of IJCAI 2001 Workshop on Intelligent Techniques for Web Personalization*, pages: 53-60, 2001.

[17] Pedro, Bernaola-Galván, and Plamen, Ch. Ivanov, and Luís A. Nunes Amaral, and H. Eugene Stanley, Scale invariance in the nonstationarity of human heart rate. In *Phys. Rev. Lett.*, vol. 87, no. 16, pages: 168-105, 2001.

[18] Resnick, P., and Varian, H., Recommender systems. In *Communications of the ACM*, vol. 40, no. 3, pages: 56-58, 1997.

[19] Sarwar, B., Karypis, G., Konstan, J., and Riedl, John, Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web, Hong Kong, China*, pages: 285-295, 2001.

[20] Sarwar, B. M., Karypis, G., Konstan, J., and Riedl, J., Recommender systems for large-scale e-commerce: scalable neighborhood formation using clustering. In *Fifth International Conference on Computer and Information Technology*, 2002.

[21] Song, H. S., Kim, J. K., and Kim, S. H., Mining the change of customer behavior in an internet shopping mall. In *Expert Systems with Applications*, vol. 21, no. 3, pages 157-168, 2001.

[22] Wang, H., Fan, W., Yu, P. S., and Han, J., Mining concept-drifting data streams using ensemble classifers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington, D.C., USA*, pages: 226-235, 2003.

[23] Widyantoro, D. H., Ioerger, T. R., Yen, J., Tracking changes in user interests with a few relevance judgments. In *Proceedings of the 12th international conference on Information and knowledge management, New Orleans, LA, USA*, pages: 548-551, 2003.

[24] Zhang, Y., Callan, J., and Minka, T., Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, Tampere, Finland*, pages:81-88, 2002.

[25] Ziegler, C. N., McNee, S. M., Konstan, J. A., and Georg, Lausen, Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web, Chiba, Japan*, pages: 22-32, 2005.