Knowledge-Based Systems 74 (2015) 14–27

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Leveraging multiviews of trust and similarity to enhance clustering-based recommender systems

Guibing Guo [a,*], Jie Zhang [a], Neil Yorke-Smith [b,c]

[a] School of Computer Engineering, Nanyang Technological University, Singapore
[b] American University of Beirut, Beirut, Lebanon
[c] University of Cambridge, Cambridge, UK

ARTICLE INFO

Article history:
Received 15 April 2014
Received in revised form 25 September 2014
Accepted 28 October 2014
Available online 4 November 2014

Keywords:
Recommender systems
Multiview clustering
Collaborative filtering
Cold start
Similarity
Trust

ABSTRACT

Although demonstrated to be efficient and scalable to large-scale data sets, clustering-based recommender systems suffer from relatively low accuracy and coverage. To address these issues, we develop a multiview clustering method through which users are iteratively clustered from the views of both rating patterns and social trust relationships. To accommodate users who appear in two different clusters simultaneously, we employ a support vector regression model to determine a prediction for a given item, based on user-, item- and prediction-related features. To accommodate (cold) users who cannot be clustered due to insufficient data, we propose a probabilistic method to derive a prediction from the views of both ratings and trust relationships. Experimental results on three real-world data sets demonstrate that our approach can effectively improve both the accuracy and coverage of recommendations as well as in the cold start situation, moving clustering-based recommender systems closer towards practical use.

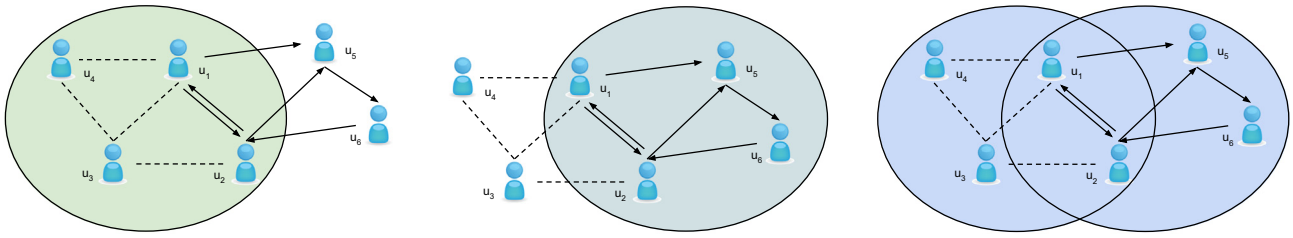
1. Introduction

Collaborative filtering (CF) [1] is a widely-exploited technique in recommender systems to provide users with items that well suit their preferences. The basic idea is that a prediction for a given item can be generated by aggregating the opinions (i.e., ratings) of like-minded users, i.e., the users with similar interest. CF has been extensively studied over decades, and many approaches [1,15,10] have been proposed in the literature. These approaches can be classified into two categories: *memory-based* and *model-based* methods. Memory-based methods [1,10] aim to find similar users (called nearest neighbors) by searching the entire user space, that is, the similarity between each user and the active user (who desires recommendations) needs to be computed using some similarity measure such as the Pearson correlation coefficient [1]. Although CF gained popularity due to its simplicity, the time-consuming procedure of searching for similar users poses a big challenge when facing large-scale data sets, which is a typical characteristic of Web 2.0. Other issues of memory-based methods include *data sparsity* and *cold start* problems [10] since the computed similarity may not be reliable due to insufficient ratings.

In contrast, model-based methods (e.g., [30,31]) can address these issues by training a prediction model offline using all the rating data (both relevant and irrelevant to the active user) rather than only based on the overlapping ratings between users. Among the various approaches, matrix factorization [15] is arguably the most popular model-based technique. It factorizes the user-item rating matrix into small ranks of user-feature and item-feature matrices. Then, the prediction is generated by the inner product of a user's feature vector and an item's feature vector. Generally, these methods can well adapt to large-scale data sets and cope with the data sparsity problem. However, a critical drawback is that the newly-issued ratings cannot be quickly involved for predictions: retraining a model is usually time-consuming and costly. This is a drawback because millions of new ratings may be reported every few hours in real applications. In addition, a lesson learned from the Netflix competition shows that the best method is a combination of hundreds of different recommendation algorithms, and none of a single algorithm can achieve the best performance over the others [2]. In this regard, it is still meaningful to develop other kinds of model-based methods. In this work, we focus on the development of a clustering-based approach based on both user ratings and trust information. In this article,

* Corresponding author.
E-mail addresses: gguo1@ntu.edu.sg (G. Guo), zhangj@ntu.edu.sg (J. Zhang), nysmith@aub.edu.lb (N. Yorke-Smith).

http://dx.doi.org/10.1016/j.knosys.2014.10.016
0950-7051/



(a) Similarity Clustering (b) Trust Clustering (c) Multiview Clustering

Fig. 1. The user clustering approaches based on similarity and trust information. Circles denote the formed user groups (clusters), and dashed lines indicate the similarity between two users, while solid lines with arrows represents user trust (social trust is directional): (a) clustering users by similarity; (b) clustering users by trust and (c) clustering users by both similarity and trust (i.e., multiviews) where more users can be grouped in one cluster.

we adopt the definition of trust as "one's belief towards the ability of others in providing valuable ratings" given by Guo [9]. By definition, trust has a much stronger correlation with user preferences than other general social connections (e.g., friendship).

Clustering-based approaches [22] offer an alternative to model-based methods. Instead of decomposing the rating matrix into matrices with small ranks, these approaches reduce the search space by clustering similar users or items together. For example, as illustrated in Fig. 1, users can be clustered by either similarity (a) or trust (b) such that the search space for nearest neighbors can be effectively narrowed down (to smaller clusters). In this way, new ratings of clustered users or items can be timely made use of to make predictions. However, clustering-based methods have not been widely exploited in recommender systems. Although demonstrated to be efficient and scalable to large-scale data sets, they are recognized to suffer from relatively low accuracy and coverage [22,28,3]. This is mainly because similar users can only be selected from the fixed size of cluster members, and in general a fewer number of similar users can be identified (than searching the whole space). In addition, the recommendation performance is also sensitive to the quality of the clustering methods. As a consequence, relatively low accuracy and coverage are observed, and these issues severely hinder the practical use of clustering-based approaches in recommender systems. To sum up, as dimension reduction models, clustering-based approaches retain the advantages of low computational cost (for searching candidate users) over memory-based approaches, and are capable of integrating newly-issued ratings for up-to-date recommendations relative to matrix factorization-based models. However, clustering-based approaches are less exploited in the literature, and suffer from relatively low accuracy and coverage.

To cope with the aforementioned issues, we develop a multiview clustering method by making use of both the view of rating patterns and the view of social trust relationships. Specifically, users are iteratively clustered from the two views using a classic partitional clustering algorithm, and clusters generated from different views are then combined together (e.g., as illustrated in Fig. 1(c)). The underlying assumption is that similarity and trust provide different views of user correlations.

Multiview-based clustering methods have not been well exploited in recommender systems and most previous works only function in a single view, namely, the user similarity. The proposed multiview clustering method has several advantages relative to single view clustering methods. First, since the clusters of users from different views will be integrated together, there are more candidate users from which similar users can be identified. Hence intuitively, both the recommendation accuracy and coverage will be improved, as we will demonstrate. Second, to accommodate users who appear in two different clusters simultaneously, we employ a support vector regression (SVR) model [7] to determine a proper prediction for a given item based on user-, item- and prediction-related features, described in Section 4.2. By doing so, the recommendation performance can be further improved. Third, to accommodate (cold) users[1] who cannot be clustered due to insufficient data, we propose a probabilistic method in Section 5 to derive a prediction from the viewpoints of both ratings and trust relationships. A series of experiments are conducted in Section 6 based on three real-world data sets, namely Flixster, FilmTrust and Epinions. The results confirm that our approach can effectively improve both the accuracy and coverage in comparison with the other counterparts, and function significantly better in handling cold users than trivial strategies (such as the average of all cluster predictions) used in previous approaches.

In summary, the main contributions of this article are:

1. We propose a multiview clustering method to cluster users from both the views of user similarity and trust. To our best knowledge, we are the first to propose a multiview clustering method based on both kinds of information.
2. We propose a support vector regression (SVR) model to handle the situation where two predictions are generated from two clusters. A number of user-, item-, prediction-related features are identified for this purpose.
3. We propose a probabilistic method to resolve the cold start problem that has not been addressed previously. Both ratings and trust information are adopted in the method.
4. We conduct a series of experiments on three real-world data sets to verify the effectiveness of the proposed multiview clustering method in comparison with other methods.

Our work takes the first step to cluster users from multiple different views of user preference rather than a single view, and verifies the ability to mitigate the issues of low accuracy and coverage using real-world data sets, moving clustering-based recommender systems closer towards practical use.

The rest of this article is organized as follows. Section 2 gives an overview of the related research on trust-based and clustering-based recommender systems. Then, our approach is elaborated in the threefold: formulating the multiview clustering algorithm in Section 3, generating predictions by support vector regression in Section 4, and handling the cold start problem in Section 5. After that, experiments based on three real-world data sets are conducted in Section 6. Finally, Section 7 concludes our present work and outlines the future research.

[1] The cold-start or cold users refer to those who rated only zero or a small number of items, e.g., less than 5 items.

2. Related work

Trust has been extensively studied in recommender systems, working as an additional dimension to help model user preference. The principle is that people trusting each other are likely to share similar preferences [24]. Trust is often investigated in memory-based methods. For example, Massa and Avesani [18] build a trust-aware recommender system by replacing user similarity with trust that can be propagated in the trust networks. They find that trust is able to mitigate the issues of traditional CF such as data sparsity and cold start. However, the more common usage of trust is to combine it with similarity in CF. For example, Guo et al. [10] merge the ratings of trusted neighbors in order to form a new and more complete rating profile for the active users based on which recommendations can be generated by integrating similarity and trust into CF. In addition, trust is also used in model-based methods. Ma et al. [17] propose a *social trust ensemble* (STE) method, which linearly combines a basic matrix factorization approach and a social trust based approach. This approach is further enhanced by Jamali and Ester [13] where trust propagation is enabled in the social networks. Recently, Yang et al. [29] propose a TrustMF method to consider both the influence of trustors and trustees[2] in a matrix factorization method. They show that better performance can be obtained using the new model. In conclusion, trust-aware recommendations can improve the performance of rating-based recommender systems, indicating that trust is able to provide an effective view of user preference in addition to similarity.

On the other hand, clustering-based approaches gain less attention in recommender systems although being demonstrated to be efficient and scalable to large-scale data sets [22,28]. As a dimension-reduction method, they are capable of alleviating the sparsity of rating data [21]. Most previous works focus on clustering users or items from the view of similarity. For example, Sarwar et al. [22] apply the bisecting k-means algorithm to cluster users and base the neighborhood formation on the cluster members. However, they find that the accuracy is decreased around 5% in comparison with the conventional kNN CF method. Xue et al. [28] show that close accuracy can be obtained at the expense of rating coverage. Recent works such as Bellogìn and Parapar [3] report that by applying more advanced clustering method, the accuracy can be further improved and even outperform the other CF approaches. However, coverage remains an unresolved issue. In summary, previous clustering-based approaches suffer from relatively low accuracy and, especially, coverage. This motivates us to develop a better clustering method that is capable of alleviating these issues.

Few works have attempted to incorporate social relationships into clustering-based methods with the aim of better performance of CF. DuBois et al. [8] combine a correlation clustering algorithm and trust models together to derive trust from the connection distance in a trust network. However, only limited improvement (around 0.0001 in mean absolute error) is observed, and their approach requires the numerical trust values which are not available in all the existing (to our best knowledge) publicly available recommendation datasets. Another drawback of the existing approaches is that most of them do not take care of the cold start problem. In their experiments, they either simply remove cold users from data sets or adopt the average value as the prediction. In this work, we presume that similarity and trust are conditionally independent characteristics (attributes) of user preference, and hence users can be clustered from both views of similarity and trust rather than merging them into a single view. In addition, we develop a probabilistic method to resolve the cold start problem based on both ratings and trust information. Thus, we open a new way to cluster users, i.e., from multiple different views of user preferences.

Multiview methods have been studied only in a very limited manner in recommender systems. Oufaida and Nouali [20] propose a recommendation method that hybrids the recommendations derived from multiple views, including collaborative, social and semantic views. Rather than to generate recommendation separately, our approach aims to produce individual recommendations by properly integrating different kinds of information. The most related work is published by Li and Murata [16], where a multidimensional clustering method is proposed to cluster users separately according to different subsets of item attributes. They aim to improve the diversity of item recommendations by avoiding providing many similar items. In contrast, our work focuses on a more principled multiview clustering algorithm based on two user features, i.e., similarity and trust, and targets better predictive accuracy and coverage. To the authors' best knowledge, our approach is the first to form a multiview clustering method merely dependent on users' ratings and trust information.

3. Multiview clustering

We first introduce the background of the multiview clustering algorithm, and then elaborate how to apply it in recommender systems together with the k-medoids approach [5].

The multiview clustering algorithm was first introduced by Bickel and Scheer [4]. The basic idea is to seek clusterings in different subspaces of a data space, i.e., the user space in our case. Users have two different kinds of information, namely ratings issued on items of interest and trust specified on other users (e.g., friends). Hence, these types of information describe users from different views, i.e., rating patterns (user behaviors) and trust links (social connections). In this section, we aim to cluster users using both ratings and trust information.

3.1. Multiview clustering algorithm

The most well-known partitional clustering methods are the *k-means* and *k-medoids* algorithms due to their simplicity and effectiveness. The former algorithm is adopted by many works or used as a baseline approach [28,3], whereas the latter has not been used in recommender systems, to the authors' best knowledge. Since the k-means algorithm generates a cluster center (centroid) by averaging all the values of each attribute, it will eliminate important personal information such as trusted neighbors. Instead, the k-medoids algorithm chooses a real user as the centroid which minimizes the summation of pairwise distances within a cluster. Mathematically, the objective function is given as follows:

$$J = \min \sum_{c \in C} \sum_{u,v \in c} d(u, v), \tag{1}$$

where C is a set of user clusters, users u, v are members of cluster $c \in C$, and $d(u, v)$ defines the distance of users u and v. We adopt the k-medoids algorithm in order to preserve individuals' ratings and trust information during the clustering process described as follows.

First, users are clustered using the rating information. In particular, user similarity is used as the distance metric to measure the closeness of two users. For clarity, we keep symbols u, v for users

[2] Trustor refers to the users who trust others, and trustees are those who are trusted by other users.

and i, j for items, and thus denote $r_{u,i}$ as a rating reported by user u on item i. We denote I_u as the set of items rated by user u. The Pearson correlation coefficient [1] is often adopted to compute user similarity:

$$s_{u,v} = \frac{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{u,v}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{u,v}} (r_{v,i} - \bar{r}_v)^2}}, \quad (2)$$

where $s_{u,v} \in [-1, 1]$ is the similarity between users u and v, $I_{u,v} = I_u \cap I_v$ is the set of items commonly rated by both users, and $\bar{r}_u$ and $\bar{r}_v$ represent the average of ratings given by users u and v, respectively. The user distance is thus computed by $d_s(u, v) = 1 - s_{u,v}$. Based on this, the k-medoids algorithm can be applied to cluster users.

Second, separately, users are clustered using the trust information. Although a user may specify other users as trusted neighbors and indicate the extent to which they are trustworthy, generally in real applications we only get binary values of trust (i.e., trust links) due to the concerns of, for instance, privacy. This deteriorates the effectiveness of trust inference methods such as *MoleTrust* [18]. Hence, we define trust values as:

$$t_{u,v} = \frac{1}{d_{u,v}}, \quad (3)$$

where $t_{u,v} \in (0, 1]$ is the trustworthiness of user v relative to user u, and $d_{u,v}$ is the minimum distance between two users determined by a breath-first search in the trust network, where users are connected with each other by social trust relationships. The closer two users are located, the higher trustworthiness the users have. According to the theory of six-degree separation [27], any two users in the social network can be connected within a small number of steps: we thus restrict $d_{u,v} \leqslant 6$ to prevent meaningless searching. The trust distance is thus computed by $d_t(u, v) = 1 - t_{u,v}$. Based on this, the k-medoids algorithm can be applied to cluster users.

Algorithm 1. Multiview k-medoids Algorithm

Input : distance matrix D_s, D_t; cluster number k
Output: user clusters C

1 $p \leftarrow 0$;
2 randomly select k medoids m_t from trust users, $\theta_t^0 \leftarrow m_t$;
3 $C_t^0 \leftarrow v$, given $\min(d_t(v, m_t))$;
4 **while** *medoids changed* **and** < *max iterations* **do**
5 $p \leftarrow p + 1$; // in the view of similarity
6 $\theta_s^p \leftarrow \theta_t^{p-1}$;
7 swap(m_s, u), $u \in C_t^{p-1}$;
8 calculate $\text{sum}_s(u) = \sum_v d_s(u, v)$, $v \in C_t^{p-1}$;
9 **if** $\text{sum}_s(u) < \text{sum}_s(m_t)$ **then**
10 $m_s \leftarrow u$;
11 $\theta_s^p \leftarrow m_s$;
12 $C_s^p \leftarrow v$, for $\forall v$, find m_s s.t. $\min(d_s(v, m_s))$;
13 $p \leftarrow p + 1$; // in the view of trust
14 $\theta_t^p \leftarrow \theta_s^{p-1}$;
15 swap(m_t, u), $u \in C_s^{p-1}$;
16 calculate $\text{sum}_t(u) = \sum_v d_t(u, v)$, $v \in C_s^{p-1}$;
17 **if** $\text{sum}_t(u) < \text{sum}_t(m_s)$ **then**
18 $m_t \leftarrow u$;
19 $\theta_t^p \leftarrow m_t$;
20 $C_t^p \leftarrow v$, for $\forall v$, find m_s s.t. $\min(d_t(v, m_s))$;
21 **return** $C \leftarrow \text{Integrate}(C_t^p, C_s^{p-1})$;

Algorithm 2. Cluster Integration Algorithm

Input : clusters C_t and C_s, threshold θ
Output: user clusters C

1 set cluster threshold $\theta_c \leftarrow \theta$;
2 **for** *each cluster* C_t^i **in** C_t **do**
3 **if** $|C_t^i| < \theta_c$ **then**
4 $min\_dist \leftarrow \infty$;
5 $best\_id \leftarrow -1$;
6 **for** *each cluster* C_t^j **in** C_t **do**
7 $sum\_dist \leftarrow 0$;
8 $cnt \leftarrow 0$;
9 **for** *each user* u **in** C_t^i **do**
10 $sum\_dist \leftarrow sum\_dist + dist(u, m_t^j)$;
11 $cnt \leftarrow cnt + 1$;
12 $avg\_dist \leftarrow sum\_dist/cnt$;
13 **if** $min\_dist > avg\_dist$ **then**
14 $min\_dist \leftarrow avg\_dist$;
15 $best\_id \leftarrow j$;
16 **if** $best\_id > -1$ **then**
17 $C_t^j \leftarrow C_t^j \cup C_t^i$; // merging clusters
18 set $C_t^i \leftarrow \emptyset$; // pruning clusters
19 goto line 2 with C_s in place of C_t if C_s unprocessed;
20 set $C \leftarrow C_t$;
21 **for** *each cluster* C_s^j **in** C_s **do**
22 $C^i \leftarrow C^j \cup C_s^j$;
23 **return** C;

The pseudocode of our multiview clustering algorithm is presented in Algorithms 1 and 2. In Algorithm 1 the rating distance matrix D_s and the trust distance matrix D_t are taken as inputs to the multiview clustering algorithm which outputs the clusters of users. We begin with the view of trust[3] by randomly selecting k users as the initial medoids, and hence form a set θ_t^0 of trust medoids at step $p = 0$ (lines 1–2). Then each user v in the user space will be assigned to the trust medoid with whom user v has the minimum distance among all the medoids m_t. The initial user clusters in the view of trust are formed and denoted as C_t^0 (line 3). After that, the multiview clustering method will iteratively (lines 5–12 and lines 13–20) cluster users from the two different views and combine both views as the final results (line 21). In particular, during lines 5–12, we initialize the similarity medoids by the trust medoids determined in the previous step (line 6). Then they are updated by swapping each medoid with other users u within the cluster C_t^{p-1} (line 7), and by the users who achieve the minimum summation of the pairwise rating distances (lines 8–11). Lastly, user clusters C_s^p in the view of similarity are generated (line 12). Similarly in the view of trust (lines 13–20), the previously generated similarity medoids will be assigned as the initial trust medoids at step p. The trust medoids are updated in the light of trust distances, and produce a new set of user clusters C_t^p. This iterative process will continue until no medoids are changed during lines 9–11 and 17–19, or the maximum iteration number is reached (line 4). Finally, the user clusters from different views are combined together by Algorithm 2 (line 21) as the output of the multiview algorithm.

Algorithm 2 elaborates how to integrate trust and similarity clusters using merging and pruning operations. The motivation is that a cluster with few users may fail to produce reliable predictions

[3] We empirically find that there is little difference with the ordering of views in the multiview clustering.

for a given item. We observe that it is not necessary to have the exact number (k) of clusters as indicated, since the main objective is to reduce dimensionality and generate accurate recommendations. In Algorithm 2, the trust and similarity clusters are taken as input, and user clusters are obtained as output. The integration will be triggered by a criterion, i.e., the number of cluster members being less than a cluster threshold θ_c (line 1). In our case, we use the value of 5 as default value which gives good results in general.[4] For each cluster C_t^i in the clusters C_t (line 2), if the criterion is satisfied (line 3), the integration will proceed. First, we find another cluster C_t^j that achieves the minimum average distance between each member u in C_t^i and the medoid centroid m_t^j of cluster C_t^j (lines 4–15). If such a cluster is found (line 16), all the members of cluster C_t^i will be merged into cluster C_t^j (line 17). Cluster C_t^i will be pruned regardless of whether it will be merged or not (line 18). After processing trust clusters C_t, we repeat the procedure by replacing C_t with similarity clusters C_s (line 19). Finally, the clusters are combined in a pairwise manner and returned as output (lines 20–23). The pairwise combination is due to the iterative procedure where the cluster medoid is derived from the previous clusters from the other view. One advantage of cluster integration is that relatively stable and less clusters ($\leqslant k$) can be preserved even if the value of k is not indicated appropriately (e.g., too large). Thus, it can alleviate the problems of specifying an ideal value of k as input to Algorithm 1.

3.2. Complexity analysis

For each iteration of Algorithm 1, the most time-consuming parts are to iteratively search and update new similarity and trust medoids within previously generated clusters (lines 7–8 and 15–16). Specifically, the computation time is around $O(n_s^2 + n_t^2) \approx O(n^2)$, where n_s, n_t refer to the average number of members within a similarity-based and a trust-based cluster, respectively; and $n = \max\{n_s, n_t\}$. For Algorithm 2, the main computation is to identify the cluster to be merged and pruned (lines 6–15). The time complexity is $O(n_i n_j)$, where n_i, n_j refer to the average number of users in clusters C_t^i and C_t^j, respectively. As the algorithm will eliminate clusters in line 18, the whole computation time is $O(k_r n_i n_j) \approx O(n^2)$, where $n = \max\{n_i, n_j\}$, and $k_r \leqslant k$ is the number of left clusters after reduction. To sum up, the overall time complexity of the multiview clustering approach is $O(m(n^2 + n^2)) \approx O(n^2)$, where m is the maximum number of iterations. In practice, the value of m is small (around 20), and n will be far smaller than the number of total users, especially when a number (k) of clusters are generated. To boost the computation, we can adopt a parallelization technique (e.g., multi-threaded) to implement the critical time-consuming parts (e.g., lines 7–8), since there are no relations among different clusters. In this way, the time complexity can be reduced to $O(n)$, i.e., linear to the average number of cluster users.

3.3. An example

Suppose there are six users $\{u_1, u_2, u_3, u_4, u_5, u_6\}$, where the first two users are similar with each other and the same holds for the last three users. Users $\{u_2, u_4\}$ are close friends and users $\{u_1, u_5, u_6\}$ are another friend group. These relations are illustrated in Fig. 2. To start with, we randomly select three users, e.g., u_1, u_3 and u_4 as the initial trust medoids in the case of $k = 3$ clusters. Hence, user u_2 and users u_5, u_6 will be clustered and linked to users u_4 and u_1, respectively. The generated three clusters are $c_t^1 = \{u_2, u_4\}, c_t^2 = \{u_3\}$ and $c_t^3 = \{u_1, u_5, u_6\}$ at step $p = 0$, from which similarity medoids are initialized by trust medoids and then updated by swapping the medoid with any other user in the cluster

[4] Further tuning the value of θ may give better performance.

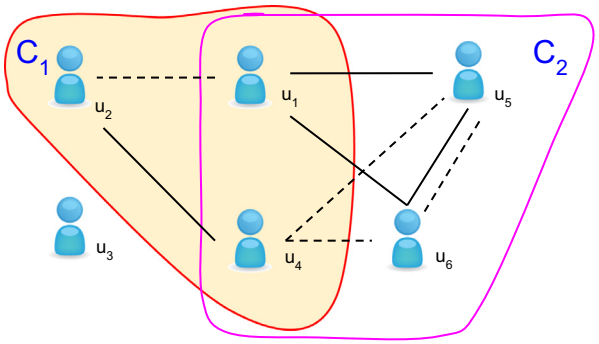


Fig. 2. An example of the multiview clustering approach. Similarity is denoted by dashed line, while mutual trust is represented by solid lines. For example, users u_1 and u_2 share similar preferences, and users u_1, u_5 and u_6 mutually trust each other. Two clusters are generated by the multiview clustering approach, namely clusters C_1 and C_2 based on both similarity and trust.

and computing the summation of pairwise rating distances within the clusters. Suppose users u_2, u_3 and u_5 are found as new similarity medoids: the user clusters based on similarity obtained are $c_s^1 = \{u_1, u_2\}, c_s^2 = \{u_3\}$ and $c_s^3 = \{u_4, u_5, u_6\}$. In the next iteration, trust medoids will be assigned by using similarity medoids initially and then updated according to trust distances. In this iteration, we note that users u_2, u_3 and u_5 are the same medoids as the last iteration and hence no update is processed. Until now we have generated stable trust and similarity clusters separately. Next, we will integrate them together. Initially, we set the threshold $\theta_c = 2$ since only a few users are available. Then the second cluster of similarity or trust will be processed with merging and pruning. Specifically, since user u_3 has no friends nor is similar to others, the clusters c_t^2 and c_s^2 will be emptied and pruned. Finally, the other clusters will be pair-wised combined together, resulting in the final clusters: $c_1 = \{u_1, u_2, u_4\}, c_2 = \emptyset$ and $c_3 = \{u_1, u_4, u_5, u_6\}$. Note that users u_1 and u_4 appear in two clusters c_1 and c_3. We will address this issue next in Section 4.

4. Prediction by support vector regression

We elaborate how to generate item predictions according to the user clusters obtained by the multiview clustering algorithm, and how to determine a proper prediction when some users appear in two different clusters due to the cluster integration.

4.1. Generating predictions

Once users are clustered by the multiview clustering algorithm, for each active user u, we may find a cluster C_u to which u belongs, and hence make an item prediction by aggregating the ratings of cluster members $v \in C_u$ who are similar to user u:

$$p_{u,j} = \frac{\sum_{v \in C_u} w_{u,v} \cdot r_{v,j}}{\sum_v w_{u,v}}, \tag{4}$$

where $p_{u,j}$ is the prediction for user u of an item j, and $w_{u,v}$ is the importance weight of user v's ratings relative to user u. Hence, more important users will have more influence on the prediction. Section 5 will handle the case where the active users cannot be clustered.

The user weight $w_{u,v}$ consists of two parts, namely similarity $s_{u,v}$ and trust $t_{u,v}$. O'Donovan and Smyth [19] suggest to use the *harmonic mean* to integrate both similarity and trust. This is because high values can only be obtained when both similarity and trust values are high. We adopt the same strategy to compute the user weight:

$$w_{u,v} = \begin{cases} \frac{2\cdot(1+s_{u,v})\cdot(1+t_{u,v})}{s_{u,v}+t_{u,v}+2} & \text{if } t_{u,v} \text{ exists,} \\ 1+s_{u,v} & \text{otherwise.} \end{cases} \quad (5)$$

Since harmonic mean requires positive values and $s_{u,v} \in [-1, 1]$, we use $1 + s_{u,v}$ instead of $s_{u,v}$ in Eq. (5), and hence $1 + t_{u,v}$ accordingly. In case that two users do not have a trust value $t_{u,v}$, we merely adopt $1 + s_{u,v}$ as user weight $w_{u,v}$ for consistency.

4.2. Prediction regression

In our multiview k-medoids algorithm, the clusters generated based on ratings and the clusters based on trust will be combined together. One resultant phenomenon is that some users may appear in two different clusters at the same time. For example, in Fig. 2 user u_4 is grouped to cluster C_1 due to her trust to user u_2 while she is also grouped to cluster C_2 due to her similarity to users u_5 and u_6. In other words, these cases happen most likely to the users who have trust connections to the other users, and who also share similar preferences with other users. According to Eq. (4), two possible predictions may be generated from different clusters. Under the assumption that users' real preferences can be well approximated using the two predictions from different clusters, we model the prediction determination as a regression problem: how to effectively combine the two predictions such that the estimated value will approximate the ground truth, i.e., user's real preference as close as possible. Note that a simpler regression (e.g., a harmonic mean) is ineffective in this case, because the two predictions may not be equally useful to determine a proper regression value. Section 6.4 will demonstrate that our approach works better than a simple average method. Formally, given that the two predictions are denoted as $p^1_{u,j}$ and $p^2_{u,j}$, we train a regression function f to map the two predictions to a value that will minimize the following loss function $J(f)$:

$$f = \min_f J(f) = \min_f \sum_{u,j} \left(f(u, j, p^1_{u,j}, p^2_{u,j}) - r_{u,j} \right)^2, \quad (6)$$

where the regression function f is associated with the active user u, target item j and two predictions $p^1_{u,j}$ and $p^2_{u,j}$. This is because even in two different cases where the two predictions are the same, e.g., 3.5 and 5.0 (suppose that ratings are ranged from 1 to 5), the ground truth for different users towards different items may differ. For example, user u may have real preference toward item j as 4.0 whereas user v prefers 5.0 in practice.

To resolve this regression problem, we apply a well-known method: support vector regression (SVR)[5] [7] stemmed from the support vector machine (SVM). SVM is a classification method for both linearly and nonlinearly separable data. It is widely applied in many applications due to its high accuracy. It always finds a global solution rather than being stuck with a local maximum. Most importantly, SVR with a proper kernel helps avoid the difficulty of using linear functions in high dimensional feature space. In particular, we use the *Gaussian radial basis function* (rbf):

$$\text{rbf}: exp(-\gamma |\boldsymbol{x_i} - \boldsymbol{x_j}|^2), \quad \gamma > 0 \quad (7)$$

as the kernel,[6] where $\mathbf{x_i}$ and $\mathbf{x_j}$ are two training examples, and parameter γ defines how much influence a single training example has. Larger γ indicates the closer the other training examples to be affected. Another parameter of SVM methods is C which reflects the tradeoff between misclassification of training examples and simplicity of the decision hyperplane. A low C makes the decision surface smooth whereas a high C tends to treat all training examples noiseless. We defer the settings of these parameters till Section 6. The features that we investigate include user-, item- and prediction-related features.

4.2.1. User-related features

Six kinds of features are identified to describe users: three are related to user behaviors (i.e., ratings) and three related to user positions within a cluster. Specifically, the former three attributes include the number, the average and standard deviation of the ratings reported by the user. These features capture the rating activity and bias of user behaviors. The latter three attributes are the rating distances to the similarity and trust medoids from the user respectively, and the rating distance between the similarity and trust medoids. These features describe the user's relative position within a specific cluster.

4.2.2. Item-related features

Ten kinds of features are identified to describe items. The first feature is the number of ratings received by the item, describing the item's popularity. The rest of the features depict the distributions of received ratings, including the (average, mode, maximum, minimum and standard deviation) of ratings, the ratio of ratings in each rating scale over all ratings, the absolute difference between mode and prediction, the number of positive and negative ratings (see definitions below). These features reflect the general opinions of users in the community.

4.2.3. Prediction-related features

Ten kinds of features are identified to represent the item predictions, where nine features regard the generation of the predictions, and the last feature is associated with the quality of predictions. Specifically, the nine features are the number and standard deviation of collected ratings, the average, maximum, minimum and standard deviation of user weights, the weighted average of user ratings (i.e., the prediction value), and the number of users identified from similarity cluster and trust cluster respectively. The first seven attributes are directly related with the generation of predictions whereas the last two attributes may help distinguish the composition of the similar users.

One more feature regarding the quality of predictions is the rating certainty [26], considering both the number of ratings involved and the conflicts between positive and negative opinions. The intuition is that the more ratings are aggregated and the higher consistency among these ratings, the more certain that the prediction is correct. Formally, Wang and Singh [26] define the certainty as follows:

$$c_{u,j} = \frac{1}{2} \int_0^1 \left| \frac{x^{m_{u,j}} (1-x)^{n_{u,j}}}{\int_0^1 x^{m_{u,j}} (1-x)^{n_{u,j}} dx} - 1 \right| dx, \quad (8)$$

where $c_{u,j} \in (0, 1]$ is the certainty of prediction $p_{u,j}$, modeled as a function of $m_{u,j}$ and $n_{u,j}$ referring to the number of positive and negative opinions (ratings) provided by the similar users regarding target item j, respectively; x is the probability of a rating being positive. We denote a rating as positive if its value is greater than the median rating scale; otherwise it is negative.

In all, we have identified 26 kinds of features regarding each user-item prediction, summarized in Table 1. By linking pairwise features together [25], we double the number of features. Since the two predictions are orderless, during the training stage we also exchange the orders of the two predictions and hence gain a new training example.

[5] The SVR used in the article is implemented by a Python module *sklearn* (http://scikit-learn.org/stable/).

[6] Generally, the rbf kernel is able to achieve good performance [23]. Although its time consumption is much higher than the linear kernel, this issue is not critical in our case since a relatively small number of features are used.

Table 1
The features that we use to represent the user-item predictions for the SVR training.

User-related features	Item-related features	Prediction-related features
1. Number of ratings reported by active user u	1. Number of ratings received on target item j	1. Number of ratings given by the similar users
2. Average of ratings	2. Average of ratings	2. Average of user weights
3. Standard deviation of ratings	3. Mode of ratings	3. Maximum of weights
4. Rating distance $d_s(m_s, u)$	4. Maximum of ratings	4. Minimum of weights
5. Rating distance $d_s(m_t, u)$	5. Minimum of ratings	5. Standard deviation of weights
6. Rating distance $d_s(m_s, m_t)$	6. Number of positive ratings	6. Standard deviation of ratings
	7. Number of negative ratings	7. Prediction value
	8. Standard deviation of ratings	8. Number of similar users from C_s
	9. Absolute difference between mode and prediction	9. Number of similar users from C_t
	10. Ratios of ratings in each rating scale over all ratings	10. Certainty of prediction $c_{u,j}$

5. Handling cold users

A known drawback of traditional clustering-based recommender systems is the inability to deal with cold users. This is because user-cluster correlations cannot be reliably computed based on few item ratings shared by cold users [12]. In fact, many existing works ignore such a case when evaluating their clustering-based recommender systems, or simply adopt the average predictions from all the clusters. In this article, we propose a probabilistic approach to identify the likelihood that a user is affiliated with a certain cluster based on both ratings and trust information.

5.1. Rating-based cluster likelihood

A rating-based likelihood is derived from the ratings reported by the cold users. Our method is based on the assumption: the ratings (of a specific item) given by users within the same cluster follow a Gaussian distribution, since users in the same cluster tend to have similar preferences, i.e., close ratings on the items commonly rated. Hence, we classify a user as an anomaly to a cluster if the average of his/her ratings does not follow the rating distribution. Specifically, the rating-based likelihood is calculated in the following three steps.

1. A Gaussian distribution $N(\mu_i, \delta_i)$ for each item $i \in I_u$ rated by the cold user u is fitted, according to the ratings given by the cluster users. Hence, the likelihood of user u's rating $r_{u,i}$ following such a distribution is:
$$Pr(r_{u,i}; \mu_i, \delta_i) = \frac{1}{\sqrt{2\pi\delta_i^2}} \exp\left(-\frac{(x-\mu_i)^2}{2\delta_i^2}\right), \tag{9}$$
where μ_i, δ_i are the mean and standard deviation of the ratings on item i, respectively.
2. The likelihood of belonging to cluster c is computed by the average of likelihoods over the item set I_u:
$$Pr(u; c) = \frac{1}{|I_u|} \sum_{i \in I_u} Pr(r_{u,i}; \mu_i, \delta_i). \tag{10}$$
Unlike the general density-based anomaly detection where probability product is used [6], we adopt the average of probabilities due to the fact that the number of ratings for each cold user may be different. Hence, a product value will be more biased to the users with smaller number of ratings.
3. An anomaly is detected if the likelihood is lower than a small value ϵ. Otherwise, the square value of $Pr(u; c)$ is taken to further increase the discrepancy of likelihoods in different clusters. Thus, the rating-based cluster likelihood is given by:
$$l(u; c) = \begin{cases} 0, & \text{if } Pr(u; c) < \epsilon; \\ Pr(u; c)^2, & \text{otherwise.} \end{cases} \tag{11}$$
In this work, we set $\epsilon = 0.05$ by default. The effect of ϵ will be left as future work. Nevertheless, it can be analyzed that smaller value of ϵ means a more relaxed constraint on the probability, and could involve more clusters in the prediction at the expense of a higher risk, whereas a higher value leads to smaller number of clusters for prediction but with more certainty.

5.2. Trust-based cluster likelihood

A trust-based likelihood is computed using the user trust relationships reported by the cold users. The intuition is that if the users trusted by a cold user are also trusted by a cluster, the cold user is likely to join the cluster. Specifically, the trust-based cluster likelihood is obtained as follows.

1. Represent the trusted neighbors of a cold user u as a value vector: $T_u = (t_{u,v_1}, t_{u,v_2}, \ldots, t_{u,v_m})$, where m is the number of users co-trusted by user u and cluster c. Note here we use lower case c to denote a cluster.
2. Compute the global trust of the trusted neighbors of the cold user, and represent it as a vector: $T_c = (t_{\cdot,v_1}, t_{\cdot,v_2}, \ldots, t_{\cdot,v_m})$, where $t_{\cdot,v_i}$ represents the global trust for user v_i in cluster c, derived by:
$$t_{\cdot,v_i} = \frac{1}{|U_{c,i}|} \sum_{v_j \in U_{c,i}} t_{v_j,v_i}, \tag{12}$$
where $U_{c,i}$ represents the set of users in cluster c who have a trust value with user v_i.
3. The trust-based likelihood is derived from the similarity between T_u and T_c. In cold conditions, cosine similarity is more preferred than the Pearson correlation coefficient since the former is computable when the length of vectors is less than 2. Thus the likelihood is:
$$l(u; t) = cos(T_u, T_c)^2 = \left(\frac{\sum_{j=1}^{n} t_{u,v_j} \cdot t_{\cdot,v_j}}{\sqrt{\sum_{j=1}^{n} t_{u,v_j}^2} \sqrt{\sum_{j=1}^{n} t_{\cdot,v_j}^2}}\right)^2. \tag{13}$$
As with Eq. (11), trust-based likelihood also adopts the square value to increase the discrepancy of cluster likelihood. A notable issue is that a cold user in social networks may also connect to very few trusted neighbors other than have rated only small items. In this case, both trust vectors T_u and T_c are quite short in length. Nevertheless, we are still able to compute the trust-based likelihood by Eq. (13). Therefore, the proposed approach is also applicable to such an extreme case.

5.3. Generating predictions

The possibility that a cold user belongs to a specific cluster is derived by aggregating the two kinds of cluster likelihoods:

Table 2
Summary statistics of the three real-world data sets.

Data set	Users	Items	Ratings	Trust	Density (%)
Flixster	5000	13,527	264,540	2898	0.39
FilmTrust	1508	2071	35,497	2853	1.14
Epinions	2438	25,786	49,230	2240	0.08

$$w_{u,c} = l(u;c) + \alpha \cdot l(u;t), \tag{14}$$

where $\alpha \in [0,1]$ is a parameter indicating the importance of the trust-based likelihood. In other words, we suggest that the similarity-based likelihood may be more reliable than the trust-based one, because the former likelihood is directly related with ratings whereas the latter functions more likely as an indicator. It makes sense in that trusted users may not share similar preference. Finally, a rating prediction is generated by averaging the predictions from different clusters according to the weights $w_{u,c}$, given by:

$$p_{u,j} = \frac{\sum_c w_{u,c} \cdot p_{u,j,c}}{\sum_c w_{u,c}}, \tag{15}$$

where $p_{u,j,c}$ is the prediction generated from cluster c for user u on item j, derived by the average of all the ratings given by users in cluster c. In case that all the weights $w_{u,c}$ are equal to 0 and Eq. (15) is not computable, we may regard all the weights equally and adopt the average of cluster predictions (if any) as the final prediction.

6. Evaluation

We conduct empirical experiments in order to study two main research questions: (1) whether incorporating multiple views of user correlations can improve the performance of recommendations in terms of accuracy and coverage and (2) whether our method can effectively cope with the cold users.

6.1. Data sets

Three real-world data sets are used in the experiments, namely Flixster, FilmTrust and Epinions. Flixster.com is a movie sharing and discovering website where users can report their movie ratings in the range from 0.5 to 5.0 with step 0.5. We randomly sample 5000 users from the original data set[7] as well as the user ratings and trust ratings. Note that, different from the other two data sets, the trust information in Flixster is symmetric. Similarly, FilmTrust[8] allows users to share movie ratings and explicitly specify other users as trusted neighbors. We adopt the data set provided by Guo et al. [12] where ratings are ranged from 0.5 to 4.0 with step 0.5. It contains 2853 trust ratings issued by 609 users. Epinions.com is a website in which users can express their opinions about products (such as books and software) by assigning numerical ratings (from 1 to 5 with step 1) and can indicate other users as trustworthy. The original data set is generated by Massa and Avesani [18] from which we randomly sample a subset by selecting the users who have rated at least three items. The statistics of data sets is shown in Table 2, where Epinions has the highest data sparsity (having a large number of items but receiving a small number of user ratings) and a relatively small amount of trust information.

6.2. Experimental settings

As discussed in Section 2, the only previous trust-based clustering approach [8] takes as input the numerical trust values that are not available in our data sets. No other state-of-the-art trust-based clustering approaches have been proposed to date. Multi-dimensional clustering approaches [20,16] take either semantic information or item contents as extra dimensions to cluster users, i.e., they do not use trust information. However, in our cases only user-item ratings and user-user trust information are available. In addition, most previous studies use the k-means as the basic clustering algorithm whereas we adopt the k-medoids to preserve individuals' ratings and trust information. Therefore, there is a lack of proper clustering-based approaches to have a fair comparison with. Further considering that the main purpose of our experiments is to demonstrate the effectiveness of our multiview clustering approach with respect to other single view-based approaches, we implement and compare with the following clustering-based methods:

- **KCF** is a baseline method where users are clustered according to the rating information by a k-medoids method, and item predictions are generated using similarity as user weights.
- **KCFT** is a variant of KCF method that computes user weights by the harmonic mean of similarity and trust (see Eq. (5)) for rating prediction. Note that except for KCF, all the other methods use this way to compute user weights. Both KCF and KCFT are single view clustering methods using rating patterns.
- **KTrust** is a single view clustering method using social trust information, i.e., users are clustered according to the trust distances by a k-medoids method.
- **MV** is our multiview k-medoids method that clusters users using both ratings and trust information. As mentioned in Section 4.2, two parameters need to be set, namely γ and C for the SVR model. In the experiments, the parameters are determined by varying their values in the range $[0,1]$.[9] Specifically, we apply exhaustive grid search in the sets $\{0.0, 0.1, 0.2, 0.3, 0.5, 0.7, 1.0\}$ for γ, and $C = 1.0$ (suggested by the *sklearn* module), and the best values are chosen using 5-fold cross validation on the training sample, guided by the *mean square errors* (MSE).

We apply 5-fold cross validation to evaluate the performance of each method. That is, all data sets are randomly split into five folds. At each time, the data of four folds are used as the training set and the last one as the test set. We repeat this procedure five times until all folds are tested and average the results. The performance is measured in terms of accuracy and coverage. *Mean absolute error* (MAE) and *root mean square error* (RMSE) are popular predictive metrics to measure the closeness of predictions relative to the ground truth:

$$\begin{aligned} \mathrm{MAE} &= \frac{\sum_{u,i\in\Omega} |r_{u,i} - p_{u,i}|}{|\Omega|}, \\ \mathrm{RMSE} &= \sqrt{\frac{\sum_{u,i\in\Omega} (r_{u,i} - p_{u,i})^2}{|\Omega|}}; \end{aligned} \tag{16}$$

where Ω is the set of test ratings, and $|\Omega|$ denotes the cardinality of the set Ω. Comparing with MAE, RMSE is useful to identify undesirably large errors. In general, smaller MAE and RMSE values indicate better accuracy. In addition, the *rating coverage* (RC) is usually defined as the ratio of the number (PR) of predictable ratings over that of all test ratings, given by:

$$\mathrm{RC} = \frac{\mathrm{PR}}{|\Omega|}. \tag{17}$$

Note that the performance of the k-medoids algorithm is sensitive to the number k of clusters and the initially selected medoids. Given

[7] http://www.cs.sfu.ca/sja25/personal/datasets/.
[8] http://www.librec.net/datasets.html.
[9] In case of value 0, the setting we use is $\gamma = 1/n$, where n is the number of features used for regression.

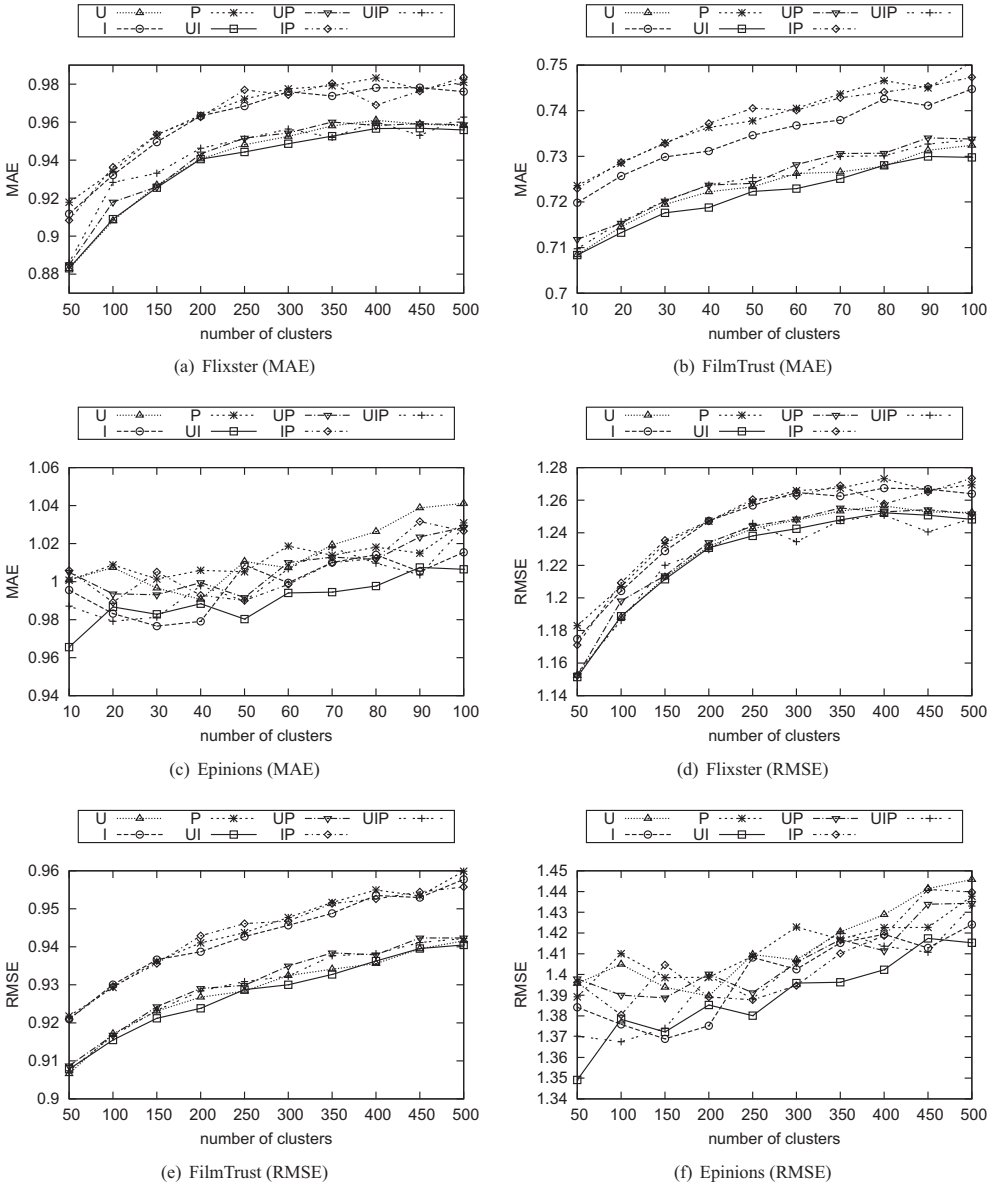
that different data sets have different statistics, especially the amount of ratings and trust information, the value of k is varied in different ranges for different data sets. More specifically, the values of k are varied from 50 to 500 with step 50 in Flixster, and from 10 to 100 with step 10 in FilmTrust and Epinions. In addition, each method is executed five times and its results averaged.

6.3. Effect of categorized features

We investigate the impact of different categories of features on the predictive performance. To facilitate the discussion, we denote U, I, P as user-, item- and prediction-based features, respectively, and concatenate these letters to represent different combinations of categorized features, e.g., UI meaning that both user- and item-related features are used whereas prediction-related features are not. The results on all the data sets are illustrated in Fig. 3. Although the differences among all these variants in performance are not significant and vary, the combination of UI achieves consistently the best accuracy on the three data sets. Since the trends on Flixster and FilmTrust are more clear and consistent, we base our conclusions on their performance. Specifically, for a single category of features, user-related features (U) perform better than item-related ones (I) which are superior to the prediction-related (P). One possible explanation is that our method can be seen as a user-based approach in distinguishing user clusters and hence determining predictions. For the combinations of categorized features, UI outperforms all the others, including all the possible features UIP. Two points can be made: (1) more features do not necessarily result in better performance, rather, noisy features (P) could decline the accuracy; (2) a proper prediction is more relevant with the nature of the active user and the target item themselves than how the prediction is generated.

6.4. Effect of the SVR model

The second series of experiments study the effect of the SVR model used in our approach, comparing with the trivial strategy of the average to determine a prediction (denoted by 'Avg'). The results on the three data sets are shown in Fig. 4. It can be seen that our method with the SVR model (MV) consistently and

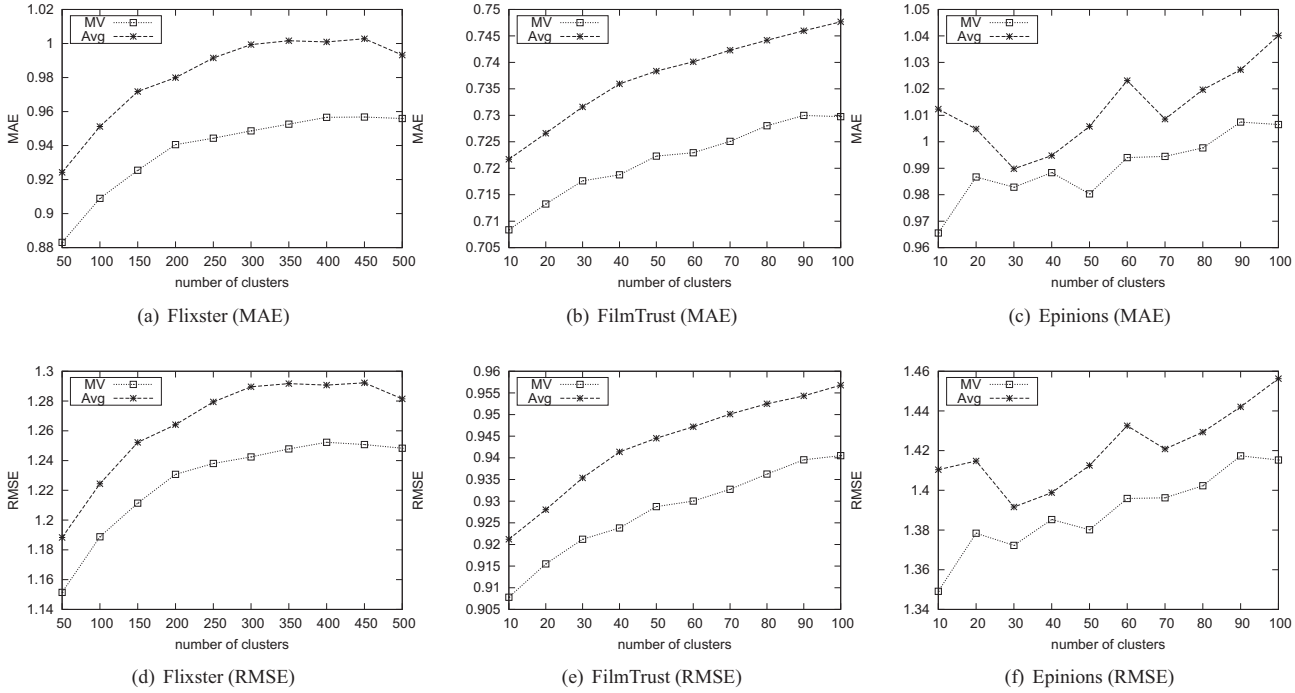


(a) Flixster (MAE) (b) FilmTrust (MAE)

(c) Epinions (MAE) (d) Flixster (RMSE)

(e) FilmTrust (RMSE) (f) Epinions (RMSE)

Fig. 3. The effect of categorized features in our approach on different data sets.



(a) Flixster (MAE) (b) FilmTrust (MAE) (c) Epinions (MAE)

(d) Flixster (RMSE) (e) FilmTrust (RMSE) (f) Epinions (RMSE)

Fig. 4. The effect of the SVR model in our approach on different data sets.

significantly perform better than the variant of 'Avg', as the number of clusters gradually increases. Hence, it is important for our method to involve the SVR model in order to achieve a proper and accurate prediction from those of two possible clusters.

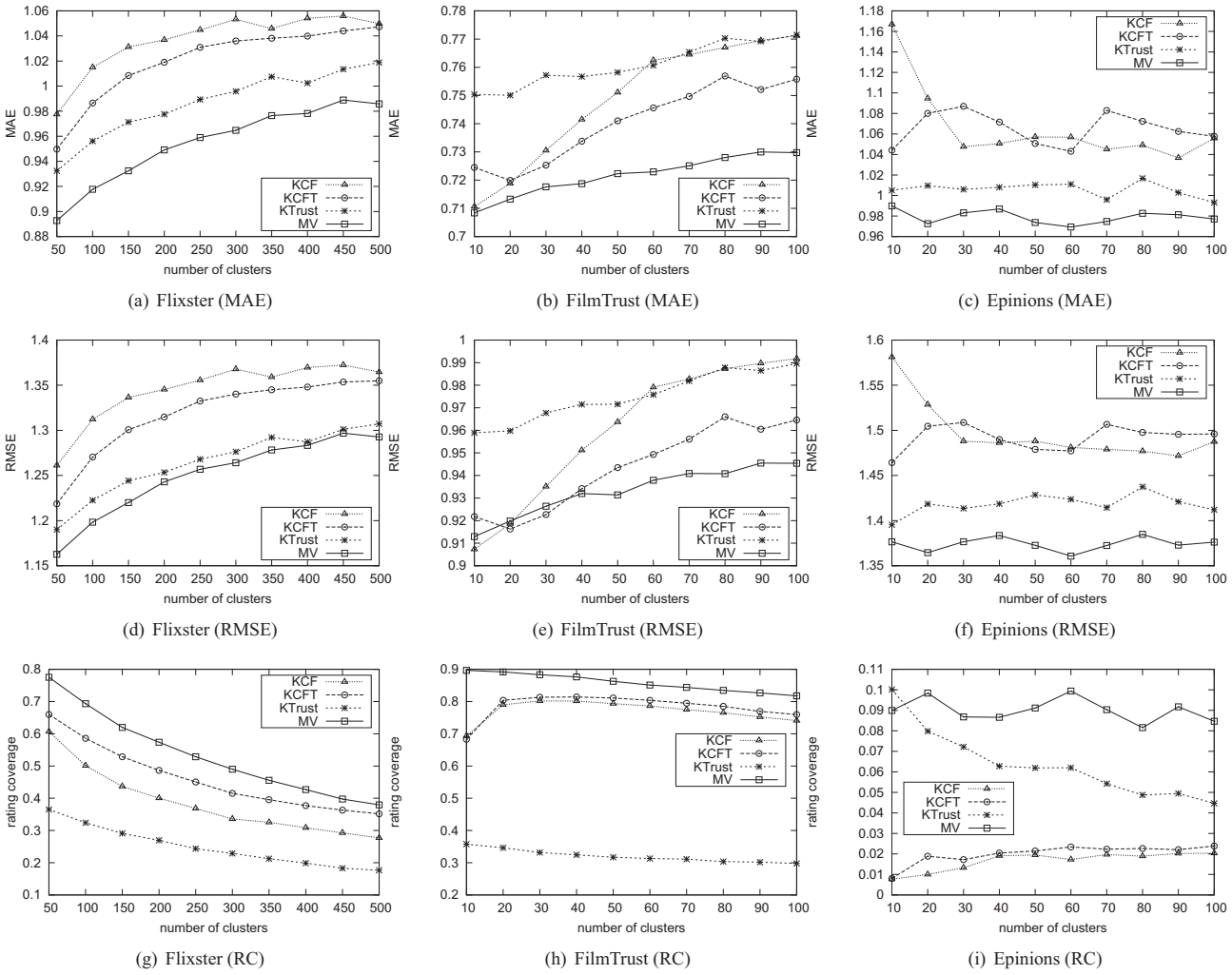
6.5. Comparison with other approaches

We apply the multiview *k*-medoids and the other methods on the three real-world data sets to investigate their predictive performance in terms of accuracy and coverage. Specifically, we take the data of users who have rated at least five items or specified at least five trusted neighbors as the training data in order to ensure reliable clustering. The performance on different data sets is illustrated in Fig. 5.

The results show that the multiview clustering method (MV) consistently achieves significant better performance than the counterparts in terms of both accuracy and rating coverage. Specifically, as the number of clusters increases, the performance on Flixster and FilmTrust is decreased accordingly. This is because less similar users can be identified within each cluster when more clusters are generated. However, on Epinions the performance is varied and tends to be relatively stable as the number of clusters increases. This may be due to the extreme sparsity of the data relative to the other data sets (see Table 2). Hence, there are few users who can be clustered and they are likely to be clustered in the same group. The performance varies because of the different initially selected k medoids in each iteration. In this regard, the performance on Epinions may indicate more about the robustness of algorithms than that in the other data sets.

The trends of accuracy of different methods are consistently observed in all data sets in terms of both MAE and RMSE. Of the methods that cluster users only based on user similarity (i.e., KCF and KCFT), KCFT generally outperforms the KCF method on Flixster and FilmTrust since the former takes into consideration trust values when predicting items' ratings whereas the latter does not. Hence, the weights of users in KCFT can be computed more accurately because only those who obtain both high similarity and trust (see Eq. (5)) will be regarded as more important users. On Epinions, the overall performance of the two methods is comparable, but KCFT tends to be more stable with less variances, especially when the number of clusters is less than 30. For the method (KTrust) that only uses trust information to cluster users, its performance varies in different data sets compared with that of KCF and KCFT. KTrust performs better than KCF and KCFT on Flixster and Epinions, but worse on FilmTrust. In other words, trust is more effective than similarity on Flixster and Epinions but not on FilmTrust. Hence, the conclusion drawn from the results is that the utility of trust may not be the same in different data sets and may depend on the distribution of the trust information. This also provides one more support for us to combine both similarity and trust to further improve the recommendation performance. As expected, this combination (MV) achieves the best accuracy in all data sets. Comparing with the second best method in each data set, the maximum improvements in accuracy are up to 0.04 (in MAE) and 0.06 (in RMSE) on all three data sets. Koren [14] points out that even small decrements of predictive errors may lead to significant improvements in real applications. Therefore, the achievements that we obtain are important. Statistically, we conduct the two-sample paired t-tests (with confidence 0.95) between the MV method and the other methods to demonstrate the significance of improvements. The *alternative* hypothesis is that *the mean of MAE (RMSE) derived by MV is less than that derived by other methods*. The results in all the data sets are illustrated in Table 3. It is observed that since all the p values are quite small in term of both MAE and RMSE, the *null* hypothesis will be rejected and the *alternative* is accepted. In other words, our approach outperforms the others, and the improvement is statistically significant (at the confidence level 0.95).

Rating coverage provides another dimension to compare the performance of different methods. It indicates the extent to which the ratings of target items are predictable. The results are presented in Fig. 5(g)–(i). Specifically, the trust-only method (KTrust) covers the least items due to the small amount of trust information relative to user ratings on Flixster and FilmTrust (see



(a) Flixster (MAE) (b) FilmTrust (MAE) (c) Epinions (MAE)

(d) Flixster (RMSE) (e) FilmTrust (RMSE) (f) Epinions (RMSE)

(g) Flixster (RC) (h) FilmTrust (RC) (i) Epinions (RC)

Fig. 5. The performance of all the methods on different data sets.

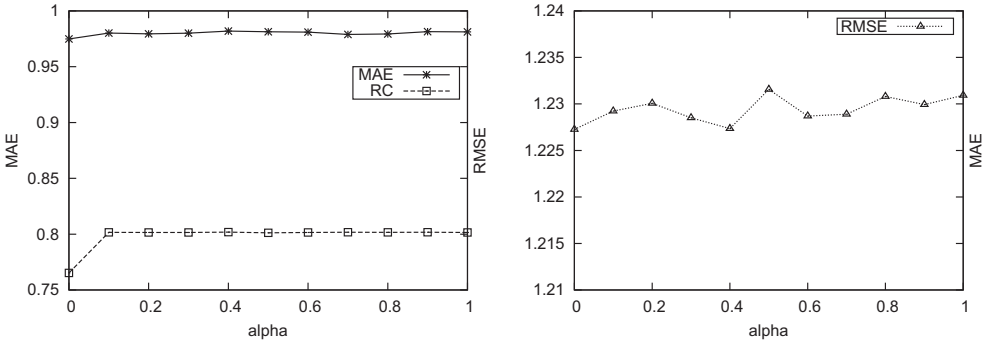
Table 3
The significance *t*-tests in all data sets.

	Methods	*df*	*t*	*p*-value
Flixster (MAE)	MV vs. KCF	9	−21.0731	$< 10^{-8}$
	MV vs. KCFT	9	−29.6059	$< 10^{-9}$
	MV vs. KTrust	9	−18.0801	$< 10^{-7}$
(RMSE)	MV vs. KCF	9	−19.3386	$< 10^{-8}$
	MV vs. KCFT	9	−25.7053	$< 10^{-9}$
	MV vs. KTrust	9	−5.6177	< 0.001
FilmTrust (MAE)	MV vs. KCF	9	−5.5937	< 0.001
	MV vs. KCFT	9	−7.9514	$< 10^{-4}$
	MV vs. KTrust	9	−55.6413	$< 10^{-12}$
(RMSE)	MV vs. KCF	9	−4.2349	<0.01
	MV vs. KCFT	9	−3.4091	< 0.01
	MV vs. KTrust	9	−44.8339	$< 10^{-11}$
Epinions (MAE)	MV vs. KCF	9	−7.4795	$< 10^{-4}$
	MV vs. KCFT	9	−16.0826	$< 10^{-7}$
	MV vs. KTrust	9	−8.7858	$< 10^{-5}$
(RMSE)	MV vs. KCF	9	−11.1419	$< 10^{-6}$
	MV vs. KCFT	9	−23.9154	$< 10^{-9}$
	MV vs. KTrust	9	−10.7601	$< 10^{-6}$

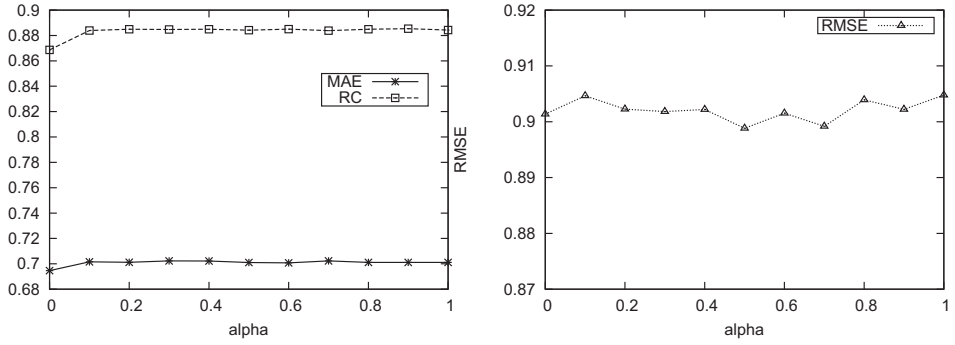
Table 2). Similarity-based approaches achieve better coverage than the KTrust method. However, in the sparse case (i.e., Epinions), trust information helps cover more items than ratings do, and as the number of clusters increases, the coverage decreases accordingly. Since user weights can also depend on trust, KCFT outperforms KCF in coverage in all data sets. Further, for the multiview clustering method (MV), since the clusters obtained from similarity and the clusters generated from trust are combined together, more users can be identified for predictions and hence more items can be recommended to the active users. Consistently, the MV method obtains the highest rating coverage and up to 20% improvements relative to the second best method on Flixster and FilmTrust and up to 5% on Epinions. In conclusion, our multiview clustering method achieves the best performance both in accuracy and coverage, comparing with other single view-based clustering methods.

6.6. Performance for cold users

We next investigate the effectiveness of our approach in dealing with cold-start users. Specifically, the users who have rated less than 5 items (including users who rated no items) in the training set are chosen and their test ratings are used as the test set. As there is no other clustering-based approach that has been proposed to handle the cold users, the most commonly adopted strategy is to take the average of predictions from all the clusters as predictions. We denote it as 'Avg' for simplicity. Note that we did not use Epinions data set because only few users (70 out of

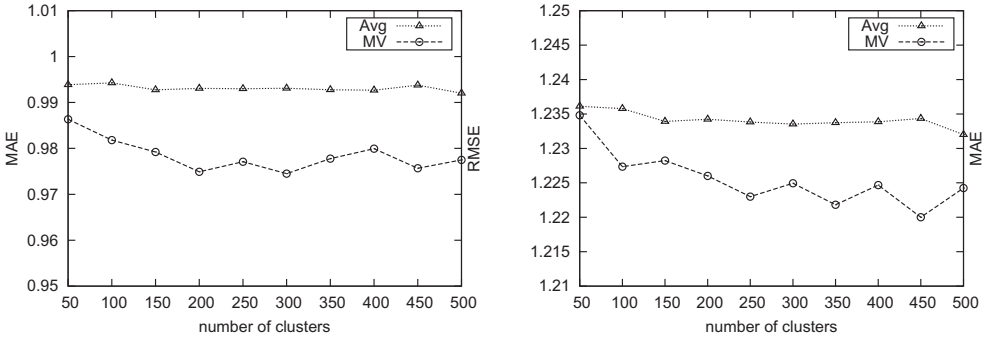


(a) Flixster (MAE, RC) (b) Flixster (RMSE)

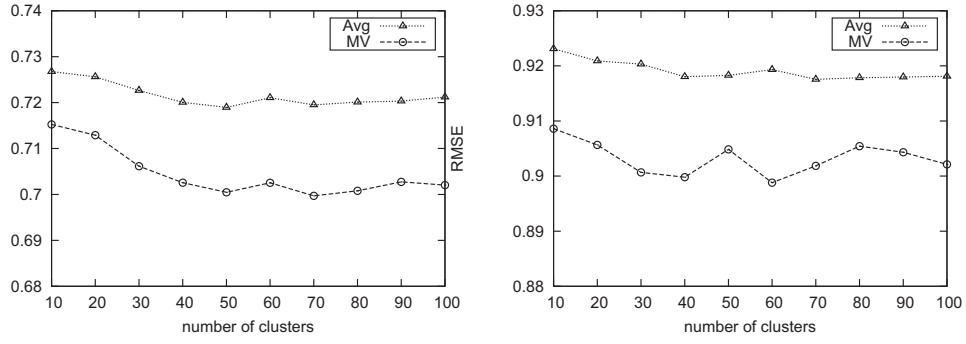


(c) FilmTrust (MAE, RC) (d) FilmTrust (RMSE)

Fig. 6. The effect of varying different values of α on Flixster and FilmTrust.



(a) Flixster (MAE) (b) Flixster (RMSE)



(c) FilmTrust (MAE) (d) FilmTrust (RMSE)

Fig. 7. The performance of our method in cold conditions on Flixster and FilmTrust.

503 test users) have trust information, and hence the resultant performance may not be representative.

6.6.1. Effect of parameter α

To select a proper value for parameter α (i.e., the importance of trust-based cluster likelihood) in Eq. (14), we first fix the number of clusters in each data set, and adjust the settings of α from 0.0 to 1.0 stepped by 0.1. The setting resulting in the best performance (both in accuracy and coverage) will be adopted for the latter experiments. More specifically, we fix $k = 100$ on Flixster, and $k = 50$ on FilmTrust. The performance of varying different values of α is illustrated in Fig. 6. The results on Flixster show that varying α values will not significantly influence the predictive accuracy both in MAE and RMSE, but have a great impact on the rating coverage when $\alpha \in [0, 0.1]$. Since α indicates the importance of trust-based cluster likelihood in determining the cluster weights, we can see that incorporating such likelihood ($\alpha > 0$) can improve the rating coverage while maintaining competitive accuracy. Specifically, we select $\alpha = 0.7$ as the best setting considering both accuracy and coverage. Similar results are observed on FilmTrust, where the best value for α is 0.6.

6.6.2. Overall performance

Adopting these settings of α, we then tune the number of clusters to show the overall performance in the cold start scenario. The results on Flixster and FilmTrust are presented in Fig. 7. It is noted that our method is capable of achieving significantly better accuracy across over the two data sets in comparison with the baseline strategy, demonstrating the effectiveness of our approach in handling cold users.

7. Conclusion and future work

Recommender systems have become ubiquitous across the web. This article proposed a multiview clustering method that clustered users both from the view of rating-based similarity and from the view of connection-based social trust, aiming to alleviate the issues of clustering-based approaches in recommender systems, i.e., the relatively low accuracy and coverage. To the authors' best knowledge, we are the first to develop a multiview clustering algorithm for recommender systems using both users' ratings and trust information. Specifically, the users were iteratively clustered according to similarity-based distances and trust-based distances by applying a classic *k*-medoids method until stable clusters were obtained. Then, the clusters generated based on similarity and the clusters generated based on trust were combined together to obtain the final clusters. A support vector regression method was employed to determine a proper prediction in the case where two predictions were generated for the users who were grouped in two different clusters due to the cluster combination. For this purpose, we proposed and identified a number of user-, item- and prediction-related features in order to describe the characteristics of user-item predictions. In addition, to accommodate the cold users who cannot be clustered and the issue of which has not been address in the previous works, we proposed a probabilistic method to identify the likelihood of belonging to each possible cluster using both ratings and trust information.

The experimental results on three real-world data sets showed that: (1) the combination of user- and item-related features were the most useful in determining a proper prediction; (2) the proposed support vector regression worked much better than a simple baseline scheme; (3) our method outperformed other approaches in term of both the accuracy and coverage; and (4) the probabilistic method can effectively handle the issue of cold-start users. To sum up, the proposed method effectively enhances clustering-based methods by virtue of the multiviews of trust and similarity, moving clustering-based recommender systems closer toward practical use.

The present work leverages trust information for users who can be connected in the trust network. One requirement of the proposed multiview clustering approach is that both user–item ratings and user–user trust information are available in the system such that users can be clustered according to different views of user preferences. However, in a general form of multiview clustering, it is possible to cluster users according to all kinds of information sources—rather than subject to ratings and social trust only—which are able to describe user preferences such as *prior ratings* [11]. In other words, the multiview clustering approach may be applicable to the situations where at least two kinds of information sources describing user preferences are available; otherwise it is not applicable.

Thus, one potential limitation to the present work is that we only consider the situations involving ratings and trust, although it may be straightforward to revise or extend it to integrate other information sources. Another limitation is that we adopt a relatively simple method to compute continuous trust values (see Eq. (3)). For future research, we intend to consider a more sophisticated trust inference approach and to consider the use of implicit trust links to enrich user trust information. It will also be interesting to empirically verify our analysis regarding the effect of ϵ (see Eq. (11)) on predictive performance. To cope with cold users, we made an assumption in Section 5.1 that ratings on a specific item given by users within a cluster follow a Gaussian distribution. For future work, we will investigate if such an assumption is valid for all the users.

Acknowledgements

Guibing Guo thanks the Institute for Media Innovation for a Ph.D. grant at Nanyang Technological University, Singapore. Neil Yorke-Smith thanks the Operations group at the Judge Business School and the fellowship at St Edmund's College, Cambridge. We also thank the anonymous reviewers for their comments which have helped improve the article from its original version.

References

[1] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. (TKDE) 17 (2005) 734–749.
[2] R.M. Bell, Y. Koren, Lessons from the netflix prize challenge, ACM SIGKDD Explor. Newsl. 9 (2007) 75–79.
[3] A. Bellogín, J. Parapar, Using graph partitioning techniques for neighbour selection in user-based collaborative filtering, in: Proceedings of the 6th ACM Conference on Recommender Systems (RecSys), 2012, pp. 213–216.
[4] S. Bickel, T. Scheffer, Multi-view clustering, in: Proceedings of the IEEE International Conference on Data Mining (ICDM), 2004.
[5] C. Bishop, et al., Pattern Recognition and Machine Learning, vol. 4, 2006.
[6] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey, ACM Comput. Surv. (CSUR) 41 (2009) 15.
[7] H. Drucker, C.J. Burges, L. Kaufman, A. Smola, V. Vapnik, Support vector regression machines, Advan. Neur. Inform. Process. Syst. (1997) 155–161.
[8] T. DuBois, J. Golbeck, J. Kleint, A. Srinivasan, Improving recommendation accuracy by clustering social networks with trust, Recomm. Syst. Soc. Web (2009) 1–8.
[9] G. Guo, Integrating trust and similarity to ameliorate the data sparsity and cold start for recommender systems, in: Proceedings of the 7th ACM Conference on Recommender Systems (RecSys), 2013.
[10] G. Guo, J. Zhang, D. Thalmann, Merging trust in collaborative filtering to alleviate data sparsity and cold start, Knowl.-Based Syst. (KBS) 57 (2014) 57–68.
[11] G. Guo, J. Zhang, D. Thalmann, N. Yorke-Smith, Prior ratings: a new information source for recommender systems in e-commerce, in: Proceedings of the 7th ACM Conference on Recommender Systems (RecSys), 2013, pp. 383–386.
[12] G. Guo, J. Zhang, N. Yorke-Smith, A novel bayesian similarity measure for recommender systems, in: Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI), 2013, pp. 2619–2625.

[13] M. Jamali, M. Ester, A matrix factorization technique with trust propagation for recommendation in social networks, in: Proceedings of the 4th ACM Conference on Recommender Systems (RecSys), 2010, pp. 135–142.
[14] Y. Koren, Factor in the neighbors: scalable and accurate collaborative filtering, ACM Trans. Knowl. Discov. Data (TKDD) 4 (2010) 1:1–1:24.
[15] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37.
[16] X. Li, T. Murata, Using multidimensional clustering based collaborative filtering approach improving recommendation diversity, in: Proceedings of the 2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012, pp. 169–174.
[17] H. Ma, I. King, M. Lyu, Learning to recommend with social trust ensemble, in: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), ACM, 2009, pp. 203–210.
[18] P. Massa, P. Avesani, Trust-aware recommender systems, in: Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys), 2007, pp. 17–24.
[19] J. O'Donovan, B. Smyth, Trust in recommender systems, in: Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI), 2005, pp. 167–174.
[20] H. Oufaida, O. Nouali, Exploiting semantic web technologies for recommender systems a multi view recommendation engine, in: Proceedings of the 7th Workshop on Intelligent Techniques for Web Personalization Recommender Systems (ITWP), 2009.
[21] M. Pham, Y. Cao, R. Klamma, M. Jarke, A clustering approach for collaborative filtering recommendation using social network analysis, J. Univ. Comp. Sci. 17 (2011) 583–604.
[22] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Recommender systems for large-scale e-commerce: scalable neighborhood formation using clustering, in: Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT), 2002, pp. 158–167.
[23] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond, the MIT Press, 2002.
[24] P. Singla, M. Richardson, Yes, there is a correlation: from social networks to personal behavior on the web, in: Proceedings of the 17th International Conference on World Wide Web (WWW), 2008, pp. 655–664.
[25] S. Vicente, C. Rother, V. Kolmogorov, Object cosegmentation, in: Proceeding of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 2217–2224.
[26] Y. Wang, M. Singh, Formal trust model for multiagent systems, in: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), 2007, pp. 1551–1556.
[27] D.J. Watts, Six Degrees: The Science of a Connected Age, WW Norton & Company, 2004.
[28] G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu, Z. Chen, Scalable collaborative filtering using cluster-based smoothing, in: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR), 2005, pp. 114–121.
[29] B. Yang, Y. Lei, D. Liu, J. Liu, Social collaborative filtering by trust, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI), 2013, pp. 2747–2753.
[30] L. Zhen, G.Q. Huang, Z. Jiang, Recommender system based on workflow, Dec. Supp. Syst. (DSS) 48 (2009) 237–245.
[31] L. Zhen, Z. Jiang, H. Song, Distributed recommender for peer-to-peer knowledge sharing, Inform. Sci. 180 (2010) 3546–3561.