Knowledge-Based Systems 67 (2014) 290–304

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Item-based top-N recommendation resilient to aggregated information revelation

Dongsheng Li [a], Qin Lv [b,*], Li Shang [a,b], Ning Gu [c,*]

[a] Tongji University, Shanghai 201804, PR China
[b] University of Colorado Boulder, Boulder, CO 80309, USA
[c] Fudan University, Shanghai 200433, PR China

ARTICLE INFO

Article history:
Received 23 June 2013
Received in revised form 17 April 2014
Accepted 25 April 2014
Available online 23 May 2014

Keywords:
Recommendation
Item-based
Collaborative filtering
Aggregated information revelation
Security

ABSTRACT

In item-based top-N recommender systems, the recommendation results are generated based on item correlation computation among all users. Therefore, recommendation results can be used to infer the correlations among recommended items. This is not an issue as long as the total amount of queries produced by a typical user is small, and the queried items among users are largely uncorrelated. However, by systematically probing the recommender system, a large amount of correlated recommendation results can be obtained and combined, and valuable aggregated knowledge, such as system-wide cross-user item popularity ranking and item clustering, can be accurately inferred. Such aggregated knowledge is of significant commercial value to online service providers, and therefore need be restricted from open access.

In this work, four aggregated knowledge attack methods are proposed to demonstrate that aggregated knowledge can be accurately inferred by attacking item-based top-N recommender systems. To make the recommender systems resilient to aggregated information revelation, a supervised randomization technique is proposed, which can protect item-based top-N recommender systems from aggregated knowledge attacks with bounded loss in recommendation accuracy. Detailed evaluation on real-world data demonstrates that the proposed attack methods can identify aggregated knowledge with high accuracy, and the proposed randomization technique can increase attack error or reduce attack precision significantly. In addition, guidelines for designing recommender systems that are resilient to such aggregated knowledge attacks are discussed in the paper.

1. Introduction

Recommender systems are widely adopted in today's mainstream online services, such as Amazon [1], YouTube [2], and Google News [3], providing useful predictions of user "ratings" or "preferences" of online items, such as products, movies, books, and news articles. Among existing recommendation techniques, item-based Collaborative Filtering (CF) is a popular method originally proposed by Amazon [1], and later adopted by other online services, such as YouTube [2]. Item-based CF methods, which leverage the idea that similar users are more likely to have similar preferences on similar items, can be applied in two types of application scenarios [4]: One is to predict item ratings, such as movie ratings on a scale of 1–5; and the other is to generate a list of recommended items. The latter is also referred to as item-based top-N recommendation.

In item-based top-N recommendation, the recommendation results are generated based on item correlation computation among all users. Therefore, recommendation results can be used to infer the correlations among recommended items. For instance, any user can query a target item and obtain a list of, typically ranked, recommended items and exam their relationship, e.g., similarities. This is not an issue as long as the total amount of queries produced by a typical user is small, and the queried items among users are mostly uncorrelated. However, as demonstrated in later sections, by systematically probing the recommender system, a large number of correlated recommendation results are obtained and combined. Valuable aggregated knowledge, such as system-wide cross-user item popularity ranking and item clustering, can be accurately inferred by malicious users.

Such system-wide aggregated knowledge is of significant commercial value to various online business applications, such as online vendor [1], e-business [13], e-service [14], trade exhibition recommendation [30], and business partner recommendation [31]. If its access is not carefully controlled, profits can be gained

* Corresponding authors. Tel.: +86 21 69589979.
E-mail addresses: qin.lv@colorado.edu (Q. Lv), ninggu@fudan.edu.cn (N. Gu).

http://dx.doi.org/10.1016/j.knosys.2014.04.038
0950-7051/

by third parties, competitors, and malicious users. For instance, competitors can develop business strategies based on such aggregated knowledge. More specifically, if an online retail provider obtains the product popularity information from Amazon, it can use such information to adjust its own online product recommendation services to improve users' shopping experience. It can further optimize its own pricing policy accordingly to improve its online sales. Furthermore, such aggregated knowledge can be leveraged by malicious users. For instance, to make a specific product be recommended more highly by the recommender system, malicious users can use the following strategy (also known as item popularity ranking attack). First, the malicious user creates multiple fake user profiles on Amazon.com, based on the knowledge of Amazon's aggregated product popularity rankings, the malicious users will identify and purchase some highly popular products together with the specific product. This tactic (named as "push" attack) has been identified and investigated by previous work [19], which shows that it can effectively make the specific product be recommended more highly. In summary, the system-wide aggregated knowledge needs to be protected, and recommender systems that are resilient to aggregated information revelation are desirable, and even essential, for the ever-increasing online services.

In this paper, four aggregated knowledge attack methods are designed and evaluated to analyze the aggregated information revelation problem in item-based top-N recommendation. Three attack algorithms, namely Naive Attack, Linear Attack and PageRank Attack, are proposed to infer item popularity ranking, and Cluster Attack is proposed to infer item clusters based on the recommendation results. The experimental results demonstrate that the proposed attack methods can predict item popularity ranking and item clusters with high accuracy. To protect against such aggregated knowledge attacks, a supervised randomization technique is proposed to obfuscate the recommendation lists with bounded recommendation accuracy loss while greatly reducing the effectiveness of aggregated knowledge attacks in item-based top-N recommendation. Detailed evaluation on a real-world dataset demonstrates that the proposed randomization technique can increase the item popularity ranking attack error and reduce the clustering attack precision substantially. To the best of our knowledge, this is the first work that analyzes and addresses the aggregated information revelation problem of item-based top-N recommender systems. The key contributions of this work are as follows:

1. The problem of aggregated information revelation in item-based top-N recommender systems is identified and analyzed. Three item popularity ranking attack methods and an item clustering attack method are proposed to demonstrate that aggregated information is indeed revealed in item-based top-N recommendation.
2. A supervised randomization technique is proposed to obfuscate the recommendation results, which can protect item-based top-N recommender systems from aggregated knowledge attacks with bounded loss in recommendation accuracy.
3. The experimental study using real-world data demonstrates that the proposed attack methods can predict item popularity ranking with relative mean average errors between approximately 8% and 30%, and cluster items with precisions between approximately 30% and 95%. Meanwhile, the proposed randomization technique can increase attack error or reduce the attack precision by at least 54%.

The rest of this paper is organized as follows: Section 2 discusses related work. Section 3 formulates the target problem via a case study on a real-world dataset. Section 4 presents in detail the four attack methods. Section 5 presents the proposed randomization technique to protect recommender systems against aggregated knowledge attacks. Section 6 presents and discusses the detailed evaluation results. In Section 7, we conclude this paper and provide guidelines for designing robust item-based top-N recommender systems.

2. Related work

Recommender systems have been an active research field [10]. Compared with content-based recommendation techniques [9], Collaborative Filtering (CF) is one of the most widely adopted recommendation techniques thanks to its high accuracy, high efficiency, and adaptability. A wide range of CF methods have been proposed, which can be classified into three categories: user-based CF [3,6], item-based CF [1,2,4], and hybrid methods [7,8,11,12]. Generally speaking, item-based CF methods achieve comparable or better recommendation accuracy than user-based CF methods [4,5] on different datasets. Item similarities are relatively more stable than user similarities [4], and item similarities can be estimated based on a subset of user ratings [1]. Therefore, item-based methods are more scalable than user-based methods, and the cold-start user problem is less of an issue in item-based recommender systems. Today, item-based CF methods have been adopted by mainstream online services, such as Amazon.com and YouTube.com.

Existing recommender systems are open, in other words, any user can obtain item recommendations, influence recommendation results [18,22], or infer certain knowledge by attacking the recommender system [26]. Several attack models have been proposed in recent years to attack collaborative filtering recommender systems [18,21,22,26]. Lam et al. proposed and analyzed the "shilling" attack models, which create a group of users (real users or agents) to give false ratings to items to mislead other users [22]. Their experimental results showed that item recommendation scores can be influenced by a group of false ratings. However, they found that the impact on item-based methods is much less than that on user-based methods. Mobasher et al. proposed a new attack model which creates a group of users with similar tastes and demonstrated that such attack model can be highly successful against item-based recommendation [18]. However, their method can only target a small set of similar items, and scales poorly for large datasets. Li et al. discovered that user privacy can be revealed by following a user's public interest in online social communities [26]. Their experimental results showed that an attacker can indeed build high correlation with a target user, and obtain private interest information of the target user from the recommendation results. Bryan et al. discovered that H_v-score is effective for detecting attack strategies [21]. Given a matrix (I,J), $H_v(I,J) = \sum_{i\in I,j\in J}(a_{ij}-a_{Ij}-a_{iJ}+a_{IJ})^2 \Big/ \sum_{i\in I,j\in J}(a_{ij}-a_{iJ})^2$, where a_{ij} is the value at position (i,j) in matrix (I,J), a_{iJ} is the mean of the i-th row, a_{Ij} is the mean of the j-th column and a_{IJ} is the mean of the whole matrix. The H_v-score can be used to detect certain types of fraudulent user profiles that show a high correlation over a subset of items in a recommender system [21]. Based on H_v-score, they proposed an unsupervised attack profile retrieval algorithm in recommender systems and showed that the proposed method performs well in distinguishing attack profiles from genuine users. However, their method is not effective enough for detecting attack profiles proposed in this paper, because malicious users only need to browse recommended items anonymously.

Since recommender systems are vulnerable to attacks, robust recommender systems which are resilient to attacks have become

an emerging research challenge during recent years [19,20,32]. Noh et al. [33,34] found that malicious users can manipulate the recommendation results in the recommender systems with fake identities (i.e., Sybils). They proposed the RobuRec method to identify fake item ratings from Sybils, so that recommendation results are only generated by honest users. Jia et al. [35] found that the recommendation accuracy of traditional recommender systems can be easily affected by malicious users. They proposed a multi-dimensional trust model to measure the credibility of user ratings on items. Using the trust model, CF methods can select reliable neighbor sets and generate recommendations with better robustness. Roy and Yan [36] found that nearest neighbor based CF algorithms are sensitive to manipulation. They observed that linear CF algorithms and asymptotically linear CF algorithms are more robust to manipulation than commonly used nearest neighbor based methods through empirical studies. Although many robust collaborative filtering methods are proposed in the literature, most of these methods focus on the case that malicious users create fake identities to manipulate the recommendation results for various purposes. Since the aggregated knowledge attack methods proposed in this paper do not rely on fake identities, existing robust CF methods cannot be directly applied to protect aggregated knowledge against such attack methods.

Randomization techniques have been adopted by existing privacy-preserving CF algorithms to protect true user ratings from being obtained by recommender systems [15–17]. Polat and Du proposed a randomized perturbation method which added noises to user item ratings before sending ratings to the recommender server [15]. This can help hide true user ratings while still achieving decent recommendation accuracy. Zhang et al. found that randomized item ratings can be derived by some learning techniques because users have the same perturbation variance [16]. They proposed a randomization technique which is guided by the recommender server. Their method discloses less private information and achieved similar recommendation accuracy, as compared with same variance perturbations. Shokri et al. proposed a distributed offline user data aggregation method which is a randomization method among different users [17]. In their method, similar users can obfuscate rating data, thus the data sent to the server is the aggregated results of many users. The randomization technique proposed in this work aims to address a different security issue, which has not been analyzed or considered by prior work. Furthermore, the randomization technique proposed in this paper focuses on top-N recommendation, and is different from the aforementioned techniques which can only be used for item rating prediction.

To the best of our knowledge, this is the first work that analyzes and addresses the aggregated information revelation problem of item-based top-N recommendation. Existing works on robust recommender systems mainly focus on the case that malicious users manipulate recommendation results using item ratings of fake identities. These methods are not applicable to protect recommender systems from aggregated knowledge attacks, because malicious users do not rely on fake identities to perform aggregated knowledge attacks in this work. Furthermore, the randomization technique proposed in this paper can protect the aggregated knowledge of top-N recommender systems, and at the same time keep the recommendation accuracy loss within predefined threshold.

3. Problem formulation

This section first presents the technical details of standard item-based top-N recommendation algorithm, and then defines the aggregated information revelation problem using a case study.

3.1. Item-based Top-N recommendation

Item-based recommendation [1] focuses on finding and recommending items that have high similarity scores to the target item. A good example of item-based top-N recommendation is the recommender system of Amazon.com, in which Amazon provides a list of recommended products based on the specific product that the user is searching and browsing. Item-based top-N recommender systems work as follows [1,4]: (1) the server computes the similarities among all item pairs; (2) recommendation scores for items are computed by taking a weighted average of a target user's ratings on items in the past; and (3) the server chooses N items with highest recommendation scores and recommend these items to the target user. An example of item-based top-N recommendation algorithm is provided later this section. Note that, the proposed work is applicable to different Item-based Top-N recommendation algorithms.

3.1.1. Item similarity computation

Given an item i, let $\vec{i} = \{r_{u_1,i}, r_{u_2,i}, \ldots, r_{u_n,i}\}$ be the vector that contains user ratings on i, and $r_{u,i}$ is user u's rating on item i. Let $I_i = \{i_1, i_2, \ldots, i_k\}$ be the set of items rated by users who also rated i, then item similarities are computed and compared between i and each item in I_i. There are different approaches to compute item similarity. One common method is the cosine similarity [1,4], which is computed as follows:

$$sim(i,j) = cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{|\vec{i}| \times |\vec{j}|} \tag{1}$$

Based on the equation above, we can obtain a set of similarity measures between i and items in $I_i : \{sim(i,i_1), sim(i,i_2), \ldots, sim(i,i_k)\}$.

3.1.2. Top-N recommendation

Given a target item i, after similarity computation with similar items, a weighted summation method [4] can be adopted to predict user u's rating on item i as follows:

$$P_{u,i} = \frac{\sum_{j\in I_i} sim(i,j) * r_{u,j}}{\sum_{j\in I_i} sim(i,j)} \tag{2}$$

If user u rated only one item $i_0 \in I_i$ in the past, then for all $i' \in I_i$, $r_{u,i'}$ is 0 except for r_{u,i_0}. Since the denominator in Eq. (2) is a constant for item i, the prediction score is totally determined by $sim(i,i_0)$. This means that if user u only rated one item i_0 in the past, then the recommended items are the items which are most similar to i_0. In this case, user u can obtain the similarity ranking of the recommendation items to i_0. For instance, if user u only clicks i_0 and is recommended with a list of similar items $\{i_1, \ldots, i_k\}$, then u can know that $sim(i_0,i_1) \geqslant sim(i_0,i_2) \geqslant \ldots \geqslant sim(i_0,i_k)$. The item similarity order information is a type of aggregated information, which reveal the correlations among items. These correlations among items can be utilized to infer other aggregated knowledge as demonstrated in later sections.

3.2. Case study

We implemented the above item-based top-N recommendation algorithm on Fudan BBS,[1] a popular online social community among Chinese universities, which has over 60,000 users, 20,000 daily posts, and 180,000 daily reads. Fudan BBS contains over 100 subcommunities, and this case study considered two of the most popular subcommunities — Graduate and News. Key characteristics of the two subcommunities are described in Table 1 in Section 6.1.1.

[1] Fudan BBS: http://bbs.fudan.edu.cn.

Table 1
Characteristics of eight subcommunities in Fudan BBS.

Subcommunity	Astrology	Auto	Basketball	Graduate	Football	Job	Joke	News
# of users	3621	1514	1693	5155	1990	3251	6494	4578
# of items	451	344	365	2047	599	386	768	1407

In this case study, the recommender system works the same as the Amazon.com recommender system [1], except that the recommended items are online articles rather than products. If a user clicks an online article, a list of recommended articles are also returned to the user. The number of articles in the recommended article list is 50, which is similar as that in Amazon.com. In the case study, malicious users work as follows: (1) click all articles and record the recommended article list of each article and (2) infer the item popularity ranking by a Naive Attack method, in which popularity of each item is measured by the number of times that item is recommended.

Fig. 1 shows the item popularity ranking attack performance of the Naive Attack method in the two subcommunities. In Fig. 1, "Real Case" means that items are ranked by their real popularity (i.e., the optimal case of item popularity ranking), "Naive Attack" means that items are ranked by the popularity obtained by the Naive Attack method, and "Random Guess" means that items are ranked by random guesses. If the item ranking of "Naive Attack" is the same as that of "Real Case", then we can conclude that all knowledge about item popularity ranking is revealed. If the item ranking of "Naive Attack" is very similar to that of "Random Guess", then we can conclude that little knowledge about item popularity ranking is revealed. As we can see from Fig. 1, item ranking of "Naive Attack" is not as perfect as "Real Case", but is much better than that of "Random Guess". The same studies are also conducted on other subcommunities of Fudan BBS, and similar results are obtained. These results indicate that "partial" knowledge about item popularity ranking is revealed in the recommendation results.

This case study demonstrates that aggregated information is indeed revealed by the recommendation results of item-based top-N recommender system, and aggregated knowledge, such as item popularity ranking, can be easily inferred by attackers. It is therefore important to design item-based top-N recommender systems which are resilient to aggregated knowledge attacks in real-world applications.

4. Aggregated knowledge attack in item-based Top-N recommendation

This section first presents how malicious users can crawl recommendation results from recommender systems. Next, four attack methods for inferring item popularity ranking and item clustering based on crawled data are proposed.

4.1. Multi-agent based crawling

Given a target item-based top-N recommender system, any user can browse online items and obtain recommendations. Similarly, malicious users can crawl item recommendations from the recommender system. However, the recommender server can identify such crawlers if the crawling strategy contains clear browsing patterns [27]. Thus, a multi-agent based crawling method is proposed to obfuscate crawling patterns among different agents, so that the recommender server cannot easily detect the crawling behavior. The basic idea of the proposed multi-agent based crawling method is to conduct a Depth First Search on the items, while the agents are randomly selected to crawl data during the search procedure. As a result, the recommender server cannot identify a clear browsing pattern from each agent. The detailed procedure of the multi-agent based crawling (MABCrawling) is presented in Algorithm 1.

Algorithm 1. MABCrawling$(A, i, visited, Q)$

Require: A is a set of agents which run on different computers and collaborate together to crawl data, and i is the target item to crawl. *visited* is a HashTable containing all visited items. $Q = \{Q_1, Q_2, \ldots, Q_{|A|}\}$, in which Q_i is the queue to store items for the i-th agent to crawl.

```
1: if visited.contains(i) == false then
2:    visited.put(i);
3:    the malicious user randomly chooses a ∈ A;
4:    Q_a.enQueue(i);
5:    the malicious user clicks i and receives a list of
      recommended items L_i;
6:    for each j ∈ L_i do
7:       MABCrawling(A, j, visited, Q);
8:    end for
9: end if
10: if all items are visited then
11:    for each a ∈ A do
12:       a crawls all items in Q_a;
13:    end for
14: end if
```

After running the MABCrawling algorithm, a malicious user can obtain a set of items, each of which is associated with a list of similar items (recommended items). Please note that, the size of the recommendation list N is an important factor which determines how much information can be gained from crawling the recommender system. N may differ in different recommender systems. For instance, N is around 4–60 in YouTube.com [2], and around 50–100 in Amazon.com as observed from its website. Smaller N values (e.g. 5 or 10) may result in poor user experience, but reveal less information. Larger N values (e.g. 100), on the other hand, may deliver good user experience, but reveal more information.

During the crawling process, if the recommender system adaptively changes its recommendation results based on the behavior of each agent, the similarity measure between the recommended items and the target item may vary. Thus, aggregated knowledge attacks cannot be easily performed. However, in the MABCrawling algorithm, each agent is first assigned a set of items to crawl by the malicious user. After assignments, each agent starts to crawl its assigned items as soon as possible. Please note that the crawling time of each agent depends on: (1) the number of items to crawl and (2) the loading time of web pages. The malicious users can increase the number of agents to ensure that each agent can finish its crawling process within a predefined time scale (e.g., 1–2 min). As a result, the recommender system is not able to update user

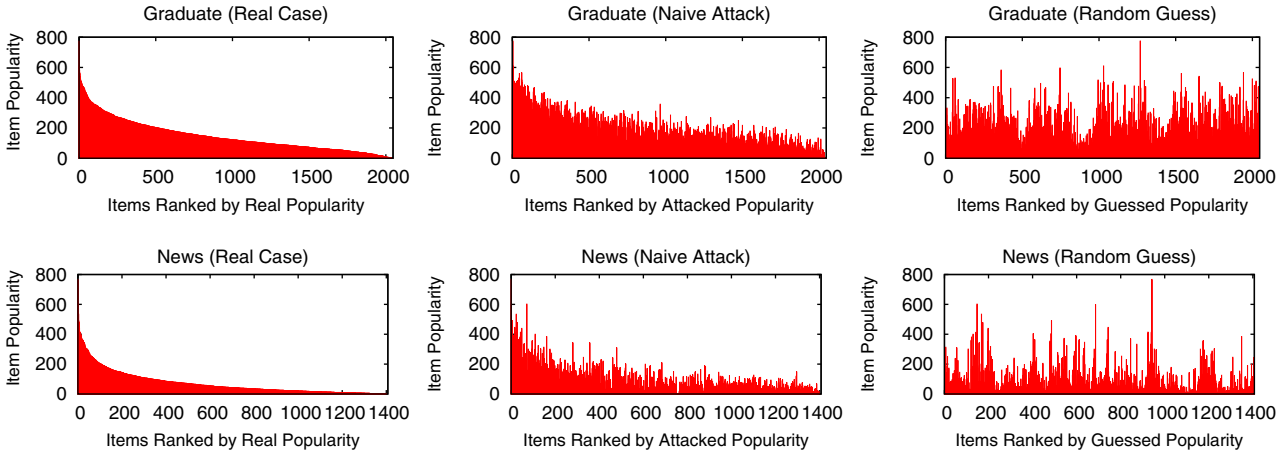

Fig. 1. Aggregated information revelation in item-based top-N recommendation: a case study on Fudan BBS.

profile and re-compute personalized recommendations for each agent in that short period of time.

4.2. Item popularity ranking attack

In item-based top-N recommender system, the crawled recommendation lists, which contain inherent relationships among items, can be utilized to infer item popularity ranking. In this section, three different attack methods are proposed to identify item popularity ranking, which have different complexities and different performance.

4.2.1. Naive Attack

The Naive Attack method exploits the observation that popular items will be recommended more frequently than unpopular items. Thus, the item popularity ranking can be estimated based on the number of times that each item is recommended. In the Naive Attack method, the popularity of each item is estimated by the following equation:

$$Popularity(i) = \sum_{j \in I} \mathbb{1}(R_j, i) \tag{3}$$

where i is the item to be estimated, R_j is the recommended item set when clicking item j. $\mathbb{1}(R_j, i)$ is an indicator function, $\mathbb{1}(R_j, i) = 1$ if $i \in R_j$, otherwise $\mathbb{1}(R_j, i) = 0$. After estimating the popularity of all items using Eq. (3), item popularity ranking can be obtained via a simple sorting process.

In the Naive Attack method, each recommendation list is checked once to compute Eq. (3), so the complexity is $O(N * m)$, where N is the size of each recommendation list (we assume that the recommender server provides the same number of recommendations to each item) and m is the total number of items. Then, a sorting algorithm is used to rank all items. The complexity of the sorting is $O(m)$ if we use the Counting Sort method. Thus, the total complexity of Naive Attack is $O(N * m) + O(m) = O(N * m)$.

4.2.2. Linear Attack

Besides the number of times an item is recommended, the Linear Attack exploits another observation that items recommended to a target item will not differ dramatically in popularity compared to the target item, i.e., similar items are very likely to have similar popularity. Thus, we propose the Linear Attack method which is the linear combination of two factors: (1) the number of times an item is recommended (as in Naive Attack) and (2) the popularity of recommended items. Then, the item popularity is estimated by the following equation:

$$Popularity(i) = \alpha \sum_{j \in I} \mathbb{1}(R_j, i) + (1 - \alpha) \frac{\sum_{j \in R_i} \left(\sum_{l \in I} \mathbb{1}(R_l, j) \right)}{|R_i|} \tag{4}$$

where $\mathbb{1}(R_j, i)$ is defined as above, R_i is the set of items that are recommended when clicking i. $\alpha \in (0, 1)$ is an empirical value, which is used to weigh the two factors in Eq. (4).

In the Linear Attack method, the same computation as Naive Attack should be performed once, and the complexity of which is $O(N * m)$. Then linear combination is performed to obtain the final popularity estimation, and the complexity is $O(N * m)$ again. Thus, the total complexity of Linear Attack is $O(N * m)$.

4.2.3. PageRank attack

PageRank [23] is the ranking algorithm used by the Google Search Engine. The basic idea of PageRank is to analyze linking relationship among web pages, which assigns a numerical weight to each element of a hyperlinked set of web pages to measure its relative importance. A web page is more important if it is linked by more links or important links. The PageRank value can be computed as follows [23]:

Step 1: For each web page p_i, the initial PageRank value $PR(p_i) = 1/M$, where M is the number of web pages.
Step 2: The PageRank values are re-computed as:

$$PR(p_i) = \frac{1 - d}{M} + d \sum_{p_j \in W_i} \frac{PR(p_j)}{L(p_j)}$$

where d is a damping factor usually set to 0.85 [23], W_i is the set of web pages that link to p_i and $L(p_j)$ is the number of outbound links of p_j.
Step 3: Repeat Step 2 until convergence is reached.

After the PageRank computation, higher PR value of a web page indicates that the web page is more important.

In item popularity ranking estimation, if items are considered as web pages and item recommendations are considered as "links" between items, then PageRank algorithm can also be applied to estimate item popularity ranking. Thus, we propose the PageRank Attack method in Algorithm 2.

Algorithm 2. PageRank Attack(I, R, d)

Require: I is a set of items, R is the set of recommendation lists that are crawled and $R_i \in R$ is the recommendation list when clicking item i. d is the damping factor.
1: $Popularity(i) = 1/|I|$;
2: **while** convergence is not reached **do**
3: **for** each $i \in I$ **do**
4: **for** each $j \in I$ **do**
5: **if** R_j contains i **then**
6: $Popularity(i) += d * Popularity(j)/|R_j|$;
7: **end if**
8: **end for**
9: $Popularity(i) += (1 - d)/|I|$;
10: **end for**
11: **end while**

In the PageRank Attack method, estimated popularity of each item is computed iteratively. Inside each iteration, the complexity is $O(N * m)$, as each item will "transfer" a share of its popularity value to its associated N recommended items. Assuming that r iterations are required to reach convergence, then the total complexity of the PageRank Attack is $O(r * N * m)$.

4.3. Item clustering attack

Cluster analysis is an important technique to understand statistical information or structure of given data, which is applied in many domains, such as social science, statistics, biology, and data mining. In cluster analysis, computing and comparing item similarities are the key operations. Since (relative) item similarities are revealed in the recommendation results, malicious users can potentially identify clusters of items based on the crawled recommendation results.

In the proposed item clustering attack method, k-centroid clustering [25], a variant of the classic k-means clustering method [24], is adopted. The basic steps of k-centroid clustering are as follows: (1) choose k items as the initial centroids; (2) for each item, compute and compare its distance (or similarity) to each centroid, and assign the item to the cluster of the centroid with the smallest distance (or largest similarity); (3) for each cluster, compute the pairwise item distances (or similarities) inside the cluster, and choose the item with the smallest average distances (or largest similarity) to other items in the same cluster as the new centroid; and (4) repeat step 2 and 3 until convergence (centroids do not change).

During the process of k-centroid clustering, the main operation is computing and comparing item distances or item similarities. Leveraging the recommendation results that are crawled, a malicious user can compare item similarities between i and each centroid if the centroids appear in i's recommendation list. But recommender systems only provide N most similar items as recommendations, which means that not all item similarities can be compared. However, we can assume that the "similar" relationships among items are transitive, i.e., if i_1 is similar to i_2 and i_2 is similar to i_3, then we can infer that i_1 is also similar to i_3. Based on this assumption, we can perform clustering attack as follows: (1) randomly choose k items as the initial centroids; (2) for each item i, if centroids are contained in R_i (R_i is the set of recommended items when clicking i), then i is assigned to the cluster of the first centroid in R_i. Otherwise, if items in R_i have been assigned to clusters, then i joins the cluster of the first assigned item in R_i; (3) repeat Step 2, until all items are assigned to clusters; (4) for each cluster c, choose the item in c that appears most times in the recommendation lists of all other items in c as the new centroid of c; and (5) repeat Steps 2–4 until centroids do not change. The details of the Clustering Attack method are presented in Algorithm 3.

Algorithm 3. Cluster Attack(I, R)

Require: I is the set of items, R is the set of recommendation lists that were crawled, and $R_i \in R$ is the recommendation list when clicking item i.
1: Randomly select k items from I as the set of initial centroids C;
2: **while** convergence is not reached **do**
3: **for** each $i \in I$ **do**
4: **for** each $j \in R_i$ **do**
5: **if** $j \in C$ **then**
6: assign i to the cluster of j;
7: break;
8: **end if**
9: **end for**
10: **end for**
11: let I' be the set of items that are not assigned;
12: **while** $I' \neq \emptyset$ **do**
13: **for** each $i \in I'$ **do**
14: **for** each $j \in R_i$ **do**
15: **if** j is assigned **then**
16: assign i to the cluster of j;
17: break;
18: **end if**
19: **end for**
20: **end for**
21: **end while**
22: for each cluster c, choose the item in c that appears most times in the recommendation lists of all other items in c as the new centroid;
23: **end while**

The complexity of the Clustering Attack method is not easy to estimate as the complexity of the second *while* loop varies from $O(m)$ to $O(m^2)$, where m is the number of items in I. Here, we consider the worst case complexity of item assignment, which is $O(m^2)$. Meanwhile, centroids re-computation takes on average $O(k) * O((m/k)^2) = O(m^2/k)$ steps, where k is the number of clusters. As k is usually a small constant, the complexity of centroids re-computation is $O(m^2)$. Assuming that r iterations are required to achieve convergence, then the total complexity of the Clustering Attack method is $O(r) * (O(m^2) + O(m^2)) = O(r * m^2)$.

5. Robust item-based recommendation

Considering the attack methods proposed in the previous section, the key information utilized by malicious users is the recommendation lists ranked based on the similarity to the items. Thus, the most effective way to protect the recommender system from these attacks is to break the similarity ordering among recommended items. This can be achieved by randomly shuffling the recommended items, after which item correlations cannot be correctly obtained by malicious users. For instance, if malicious users does not know whether one item is similar to another item, then item clustering cannot be performed.

In this section, a supervised randomization technique is proposed to achieve robust item-based top-N recommendation, which can randomize the recommendation results with bounded recommendation accuracy loss.

5.1. Impact analysis of randomization

Randomization of recommendation results may result in loss of recommendation accuracy. Thus, a key issue is how to maintain low accuracy loss during randomization. The basic randomization algorithm adopted in this paper is Fisher–Yates shuffle [29], which is described in Algorithm 4.

Algorithm 4. Shuffle(I)

Require: I is a set of items, and $|I| = m$.
1:**for** i from m to 1 **do**
2: randomly choose $j \in [1, i]$;
3: swap $I[i]$ and $I[j]$;
4: **end for**

The basic operation in Fisher–Yates shuffle is item swap, and recommendation accuracy loss occurs after each item swap. Considering items i and j, swapping i and j can affect users who like only i or only j. This is because users who like both i and j will not care about the order of i and j in the recommendation lists, and it is the same for users who like neither i nor j. Then, the recommendation accuracy loss after swapping i and j can be measured as follows:

$$\mathcal{L}(swap(i,j)) = \frac{|U_i| - |U_j|}{\sum_{u=1}^{n}\sum_{x=1}^{m} r_{u,x}} \tag{5}$$

where U_i (U_j) is the set of users who like i (j). $r_{u,x} = 1$ if user u likes item x, and $r_{u,x} = 0$ otherwise.

The goal of randomization is to break the similarity ordering among items. The positions of the swapped items have significant impacts on the "randomness" of item ordering. For instance, swapping two nearby items will have smaller impact on the ordering compared with swapping two items that are far apart. We define the increase of randomness after swapping items i and j as follows:

$$\mathcal{R}(swap(i,j)) = \#W_{after} - \#W_{before} \tag{6}$$

where $\#W_{after}$ is the number of incorrect orders after swapping and $\#W_{before}$ is the number of incorrect orders before swapping.

Based on the accuracy loss and randomness increase definitions above, swap operations with larger randomness increase and smaller accuracy loss are preferred.

5.2. Item recommendation with supervised randomization

In this paper, item recommendation is performed with the standard item-based method as described in Section 3.1. Here, we focus on the randomization technique which can permute recommendation results to prevent aggregated information revelations while keeping recommendation accuracy loss within predefined threshold.

Based on the above analysis of randomization operations, a supervised randomization is proposed to ensure largest randomness increase per accuracy loss and meanwhile keep recommendation accuracy loss within a predefined threshold. The supervised randomization method is based on Fisher–Yates shuffle, during which all item swaps are first stored. Then, the algorithm iteratively chooses swaps with the largest randomness increase/ accuracy loss ratio to perform on real recommendation results, until the accuracy loss threshold is reached. The detailed supervised randomization algorithm is presented in Algorithm 5.

Algorithm 5. Supervised Randomization(I, R, η)

Require: I is a set of items, R is the set of recommendation results. For each $i \in I$, R_i is the recommendation results when clicking i. η is the predefined threshold of accuracy loss.
1: **for** each $t \in I$ **do**
2: $R'_t = R_t$;
3: $Q_l = \emptyset$ and $Q_r = \emptyset$ are two arrays to store accuracy loss and randomness increase of each swap;
4: **for** i from $|R'_t|$ to 1 **do**
5: randomly choose $j \in [1, i]$;
6: swap $R'_t[i]$ and $R'_t[j]$;
7: $Q_l.add(\mathcal{L}(swap(i,j)))$;
8: $Q_r.add(\mathcal{R}(swap(i,j)))$;
9: **end for**
10: $\sigma = 0$. Let $\mathcal{S} = \emptyset$ be the set to store all chosen swaps;
11: **while** $\sigma \leqslant \eta$ **do**
12: choose $swap(i,j)$ with largest $\mathcal{R}(swap(i,j))/\mathcal{L}(swap(i,j))$;
13: **if** $\sigma + \mathcal{L}(swap(i,j)) \leqslant \eta$ **then**
14: $\mathcal{S}.add(swap(i,j))$;
15: $\sigma += \mathcal{L}(swap(i,j))$;
16: **end if**
17: remove $swap(i,j)$ from Q_l and Q_r;
18: **end while**
19: **for** each swap $s \in \mathcal{S}$ **do**
20: perform swap s on R_t;
21: **end for**
22: **end for**

Please note that, the total recommendation accuracy loss can be kept within η in the proposed supervised randomization. This argument is formally described and proved in Theorem 5.1.

Theorem 5.1. *Given I, R, η, the total accuracy loss $\mathcal{L}' \leqslant \eta$ after supervised randomization.*

Proof. For each item $t \in I$, R_t is randomized with the guarantee that $\sum_{s\in\mathcal{S}_t}\mathcal{L}(s) \leqslant \eta$, where $\mathcal{S}_t$ is the set of chosen swap operations after the while loop for recommendation list R_t in Algorithm 5. The total accuracy loss over all recommendation results is: $\mathcal{L}' = \frac{\sum_{t\in I}\sum_{s\in\mathcal{S}_t}\mathcal{L}(s)}{|I|}$. As $\sum_{s\in\mathcal{S}_t}\mathcal{L}(s) \leqslant \eta$, then:

$$\mathcal{L}' = \frac{\sum_{t\in I}\sum_{s\in\mathcal{S}_t}\mathcal{L}(s)}{|I|} \leqslant \frac{\sum_{t\in I}\eta}{|I|} = \eta.$$

Thus, we have $\mathcal{L}' \leqslant \eta$. □

In the supervised randomization method, the complexity of Fisher–Yates shuffle is $O(N)$, where N is the size of the recommended item list. During each swap, accuracy loss and randomness increase are calculated, and total complexity of which is $O(n)$, where n is the number of users. Then, swaps with the largest randomness increase/accuracy loss ratio are chosen with complexity of $O(N^2)$. Finally, the chosen swaps are performed on real recommendation results with complexity of $O(N)$. Thus, the total complexity of supervised randomization is $O(N) * (O(n) + O(N^2)) = O(Nn + N^3)$. As N is reasonably small in real applications (less than 100), this total complexity is acceptable.

6. Experimental results

This section evaluates the severity of aggregated information revelation of item-based top-N recommendation, and demonstrates the effectiveness of the proposed supervised randomization technique in aggregated information protection. First, a description of the dataset is presented, as well as the evaluation metrics to measure the attack performance and protection performance. Then, the proposed item popularity ranking attack methods and clustering attack method are evaluated and compared. Finally, the proposed supervised randomization is evaluated in detail.

6.1. Dataset description and evaluation metrics

6.1.1. Dataset description

The evaluation is conducted with Fudan BBS, a popular online social community among Chinese universities, which has over 60,000 users, 20,000 daily posts, and 180,000 daily reads. Eight of the most active subcommunities, which have various numbers of users and items, are considered in the experiments. More importantly, these subcommunities have diverse item popularity distributions and internal clustering structures. Together, these eight subcommunities offer a comprehensive evaluation dataset of the proposed attack methods and randomization algorithm. The dataset was collected during two consecutive weeks. The overall characteristics of the eight subcommunities are listed in Table 1.

6.1.2. Evaluation metrics

The following metrics are adopted to measure item popularity ranking accuracy, item clustering accuracy and recommendation accuracy after randomization:

1. **Item Popularity Ranking Evaluation Metric:** Relative Mean Average Error (RMAE) is typically used to measure how close estimations are to the real values relatively. RMAE is computed as follows [28]:

$$RMAE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{f_i - y_i}{n}\right| \tag{7}$$

where n is the number of items, f_i is the estimated ranking and y_i is the real ranking. Intuitively, the RMAE value means the relative ranking position error. A small RMAE value indicates high item popularity ranking estimation accuracy, and a large RMAE value indicates low accuracy.

2. **Item Clustering Evaluation Metric:** To evaluate the accuracy of item clustering, two precision metrics are adopted. The first is "total precision", which is defined as the total fraction of correctly assigned items in clustering. The second is "cluster precision", which is defined as the average clustering precision of all clusters. The two evaluation metrics are computed as follows:

$$Total\ Precision = \frac{\sum_{i\in I} A(i)}{|I|} \tag{8}$$

$$Cluster\ Precision = \frac{\sum_{c\in C}\frac{\sum_{i\in I_c} A(i)}{|c|}}{|C|} \tag{9}$$

where I is the set of items, $A(i) = 1$ if item i is assigned incorrectly and $A(i) = 0$ otherwise, and C is the set of clusters. In both item clustering evaluation metrics, a large value indicates high clustering accuracy, and a small value indicates low accuracy.

3. **Recommendation Accuracy Evaluation Metric:** In the scenario of top-N recommendation, Precision and Recall are widely adopted to measure the accuracy of recommendation results [10]. The Precision and Recall values are computed as follows:

$$Precision = \frac{|I_u \cap I_r|}{|I_r|}, \quad Recall = \frac{|I_u \cap I_r|}{|I_u|} \tag{10}$$

where I_u is the set of items that user u rated or liked and I_r is the set of items that are recommended. Higher Precision and Recall values indicate better recommendation accuracy.

4. **Item Position Shift Evaluation Metric:** The positions of recommended items are shifted in the recommendation list after randomization, and greater shift indicates better protection of aggregated knowledge. The following metric is proposed to measure how the positions of recommended items are shifted after randomization:

$$Position\ Shift = mean\{|p_i - p_i'|\} \tag{11}$$

where p_i is the position of item i in the recommendation list when i is recommended by standard item-based CF method, and p_i' is the position of item i after randomization.

6.2. Observation analysis

In the proposed item popularity ranking attack methods, two observations are adopted as the basis for inferring item popularity ranking based on crawled recommendation lists. The first observation is "popular items will be recommended more frequently than unpopular items", and the second observation is "similar items are very likely to have similar popularity". Here, we present statistical analysis results to help validate these observations.

6.2.1. Observation I

Fig. 2 shows the correlation between item popularity and the number of times that the item is recommended (N is set to 50). It shows that positive correlation between item popularity and the number of times that the item is recommended indeed exists in the two subcommunities. Meanwhile, similar results are observed in all other subcommunities. This study helps confirm that popular items will be recommended more frequently than unpopular items.

6.2.2. Observation II

Fig. 3 shows the correlation between item popularity and the average popularity of the ten most similar items to each item. We can see that most of the data points are located around the "$y = x$" line, so that linear relationship between item popularity and the average popularity of similar items to each item indeed exists in the two subcommunities. Also, similar results are observed in all other subcommunities. This study helps confirm that similar items are likely to have similar popularity.

6.3. Item popularity ranking attack performance

Fig. 4 shows the attack performance of the proposed item popularity ranking attack methods in terms of RMAE (y-axis). The x-axis N is the size of recommendation list, which determines the amount of information revelation. Please note that, we choose $\alpha = 0.1$ in the Linear Attack method, as we find that α around 0.1 achieves best attack performance in the evaluated eight subcommunities. From the results, we can see that the RMAE values range from 30% to 7.8% in different subcommunities. Among the three proposed attack methods, Linear Arrack outperforms Naive Attack and PageRank Attack in all eight subcommunities, with average RMAE reductions of 28.4% and 23.9% respectively. PageRank Attack outperforms Naive Attack in six of the eight subcommunities and achieves comparable results in the other two subcommunities with average RMAE reduction of 13.5%. Note that Naive Attack only considers the number of appearances in other items' recommendation lists. In PageRank Attack, the relationships among items are

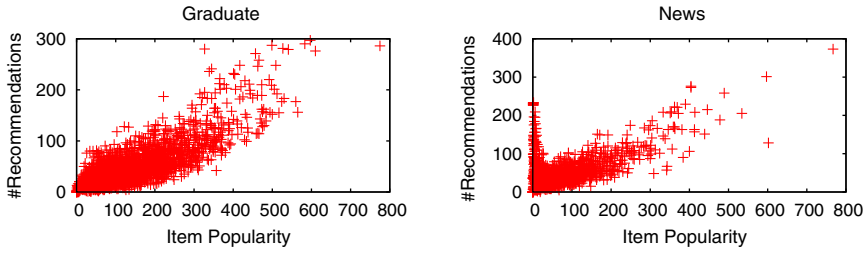

Fig. 2. Item popularity vs. the number of times the item is recommended ($N = 50$).

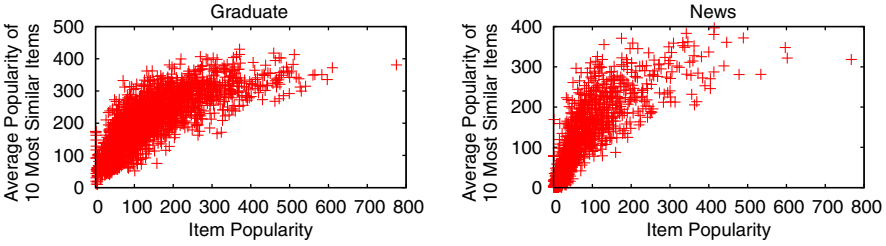

Fig. 3. Item popularity vs. the average popularity of ten most similar items to each item.

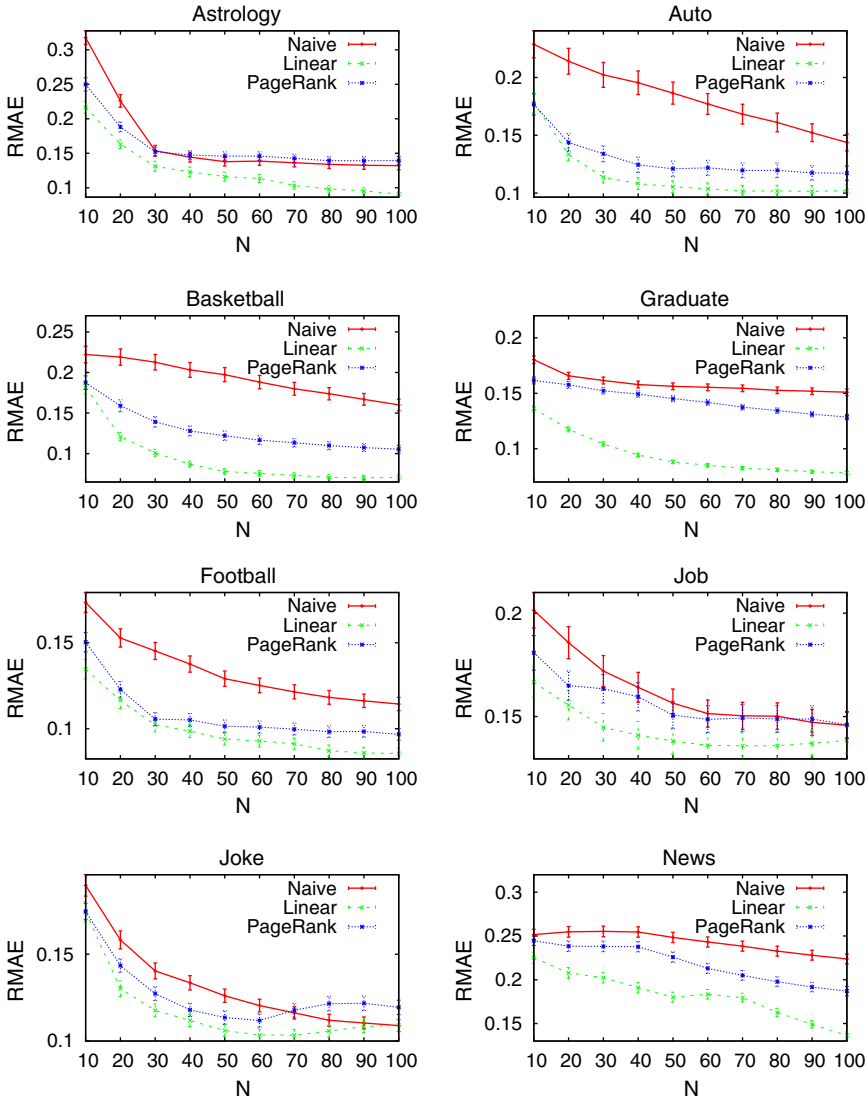

Fig. 4. Item popularity ranking attack performance in eight subcommunities.

the dominating factor for measuring the popularity of an item. In Linear Attack, the above two types of information are combined linearly, and thus achieves better attack performance.

From the experimental results, we can also see that the RMAE values decrease as N (the size of recommended item lists) increases. This is reasonable, because more information are revealed when more items are recommended. Thus, a smaller N value should be adopted to increase system robustness when designing item-based top-N recommender systems. When N is higher than 50, the RMAE values are less than 18% in all subcommunities, which indicate fairly accurate popularity ranking estimations. Thus, a N value less than 50 should be adopted in item-based top-N recommender systems.

6.4. Item clustering attack performance

Fig. 5 shows the performance of the proposed Cluster Attack method in all eight subcommunities. Please note that, in this experiment, we use the clustering results by the standard K-centroids algorithm as the true clustering. In the K-centroids clustering algorithm, the initial seed selection will have great impact on the final clustering results, thus we choose the same seeds in clustering comparisons to eliminate the influence of seeding.

We can see in Fig. 5 that the total precision values range from approximately 50% to 95% in four of the eight subcommunities with N varying from 10 to 100, from approximately 40% to 90% in two of the rest four subcommunities, and from approximately 21% to 82% in the last two subcommunities. And similar results can be observed for the clustering precisions. Also, we can see that both the total precision and the clustering precision increase as N increases, which is because larger N means more information revelations. Thus, smaller N values (less than 50, for instance) should be adopted to protect item-based top-N recommender systems from clustering attacks.

6.5. Performance of randomization

The performance of the proposed randomization technique is evaluated in two aspects: aggregated knowledge protection and recommendation accuracy loss. In the following experiments, "unsupervised" is referred to an unsupervised randomization method which works as follows: (1) recommended item lists are divided into sublists of size k'; (2) each sublist is randomized by Fisher–Yates shuffle; and (3) all sublists are reconnected as new recommendation results. The proposed supervised randomization is referred as "supervised" in the following experiments.

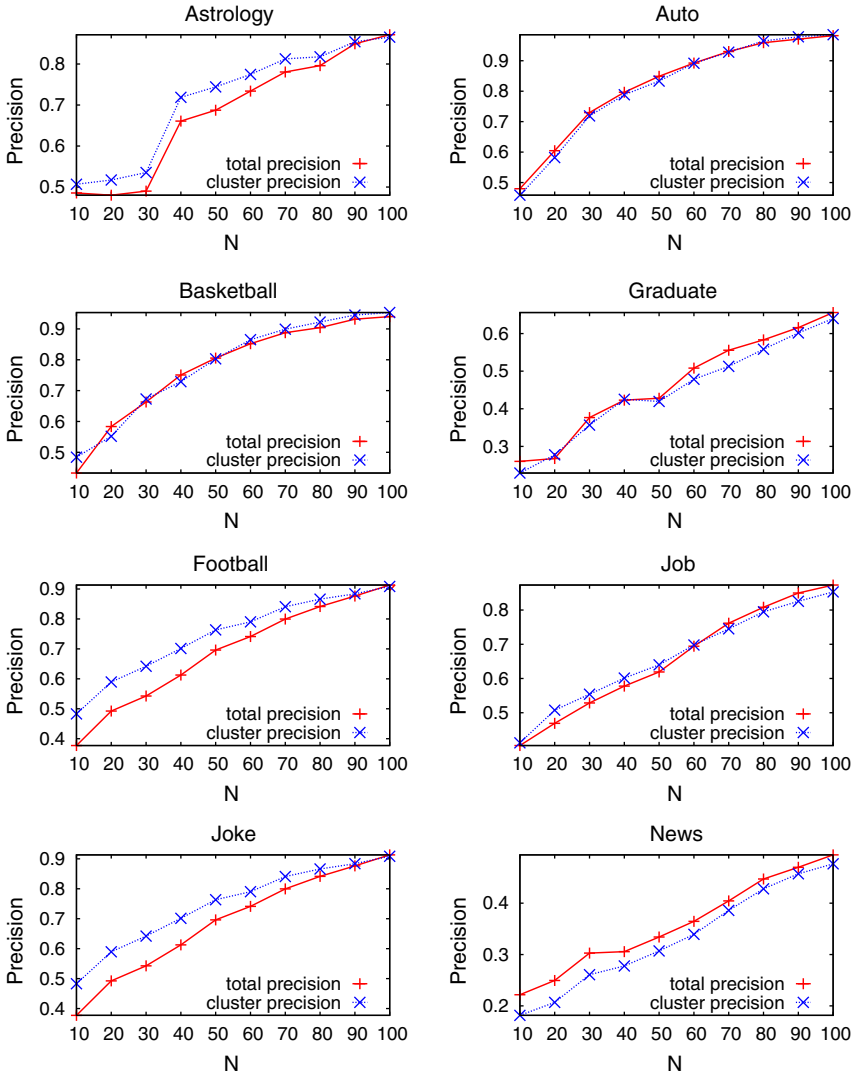

Fig. 5. Clustering attack performance in eight subcommunities.

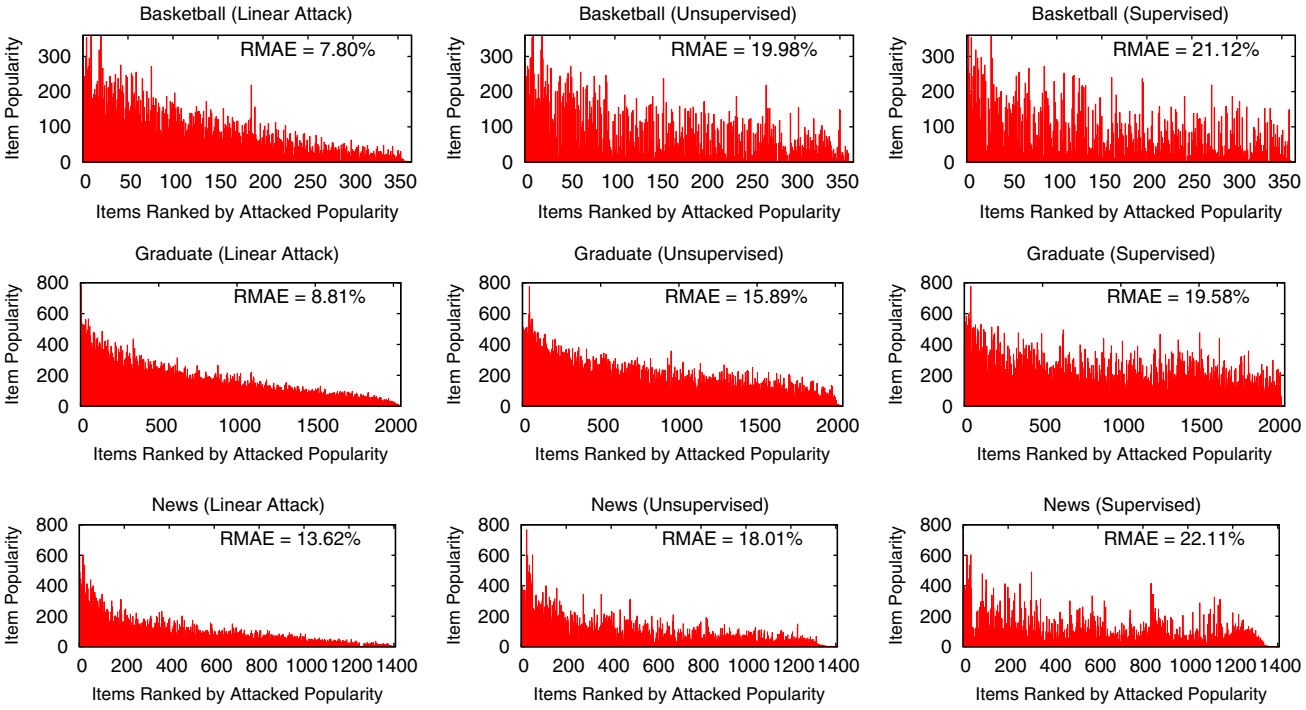

Fig. 6. Item popularity ranking attack performance of Linear Attack before and after randomizations in three subcommunities (Basketball, Graduate, News).

6.5.1. Aggregated information protection

In this experiment, we do not evaluate the performance of Naive Attack, as the randomization techniques cannot change the number of times each item is recommended, i.e., randomization will have little effect against Naive attack. But, it should be mentioned that, Naive Attack is not accurate compared with the other two item popularity ranking attack methods, and its RMAE values are similar to those of the other two methods after randomization. Thus, protections can be omitted for the Naive Attack method. In this experiment, we choose $k' = 10$ for the unsupervised method, $\eta = 5\%$ for the supervised method, and $N = 50$ for both methods. This is because the two randomization methods achieve similar recommendation accuracy loss with this setting. Thus, we can focus on comparing the performance of aggregated information protection of the two randomization methods.

Figs. 6 and 7 show the item popularity ranking performance of Linear Attack and PageRank Attack before (column 1) and after (column 2 and 3) randomization in three of the eight subcommunities.

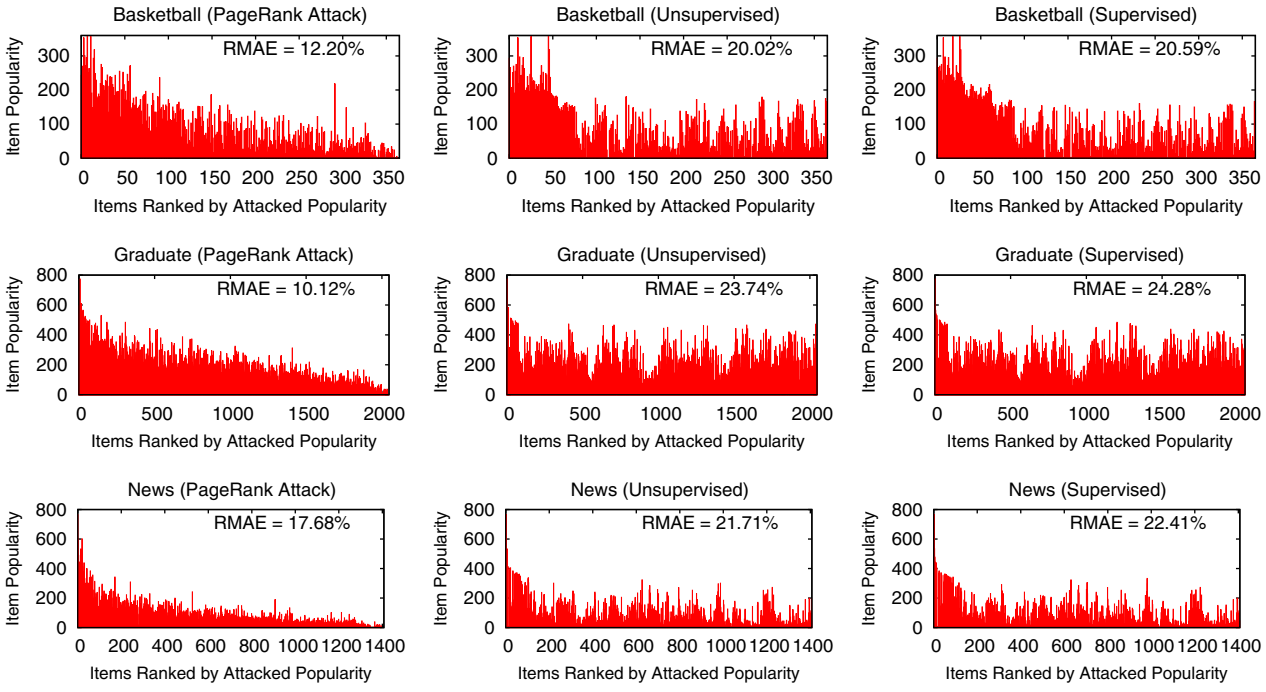

Fig. 7. Item popularity ranking attack performance of PageRank Attack before and after randomizations in three subcommunities (Basketball, Graduate, News).

Table 2
Overall performance comparison of randomization in item popularity ranking attack.

Item popularity ranking	Unsupervised		Supervised	
	Linear	PageRank	Linear	PageRank
Average RMAE gain (%)	45.7	46.5	62.4	54.8

Table 3
Overall performance comparison of randomization in cluster attack.

Cluster attack	Unsupervised (%)	Supervised (%)
Average total precision loss	63.2	66.6
Average cluster precision loss	52.6	54.8

Intuitively, we can see that the attacked item popularity rankings before randomization are better "ordered" than the attacked item popularity rankings after randomization, and the proposed supervised randomization method achieves better protection (higher RMAE) than that of unsupervised randomization in all three subcommunities. Experiments with all the other subcommunities also show similar results, and detailed statistics are listed in Table 2. The main reasons that the proposed supervised randomization method outperforms the unsupervised randomization method are: (1) items are obfuscated in wider ranges and (2) swaps with larger randomness increase can be selected by the supervised randomization method.

Table 2 shows the detailed statistics of RMAE gain of item popularity ranking attack after randomization. As shown in Eq. (7), lower RMAE value means better item popularity ranking attack performance, so that higher RMAE gain indicates better protection against item popularity ranking attack. We can see from the results that randomization techniques can significantly reduce the performance of item popularity ranking attack, and the proposed supervised randomization method can increase RMAEs of the proposed Linear Attack and PageRank Attack by 62.4% and 54.8%, respectively, on average over the eight subcommunities. The unsupervised randomization method can also reduce the performance of item popularity ranking attack, but can only increase RMAEs of the proposed Linear Attack and PageRank Attack by 45.7% and 46.5%, respectively, which are worse than the supervised randomization method.

Table 3 shows the detailed statistics of total precision loss and cluster precision loss of cluster attack after randomization. As shown in Eq. (8), higher total precision and cluster precision values mean better cluster attack performance, so that higher total precision loss and cluster precision loss indicate better protection

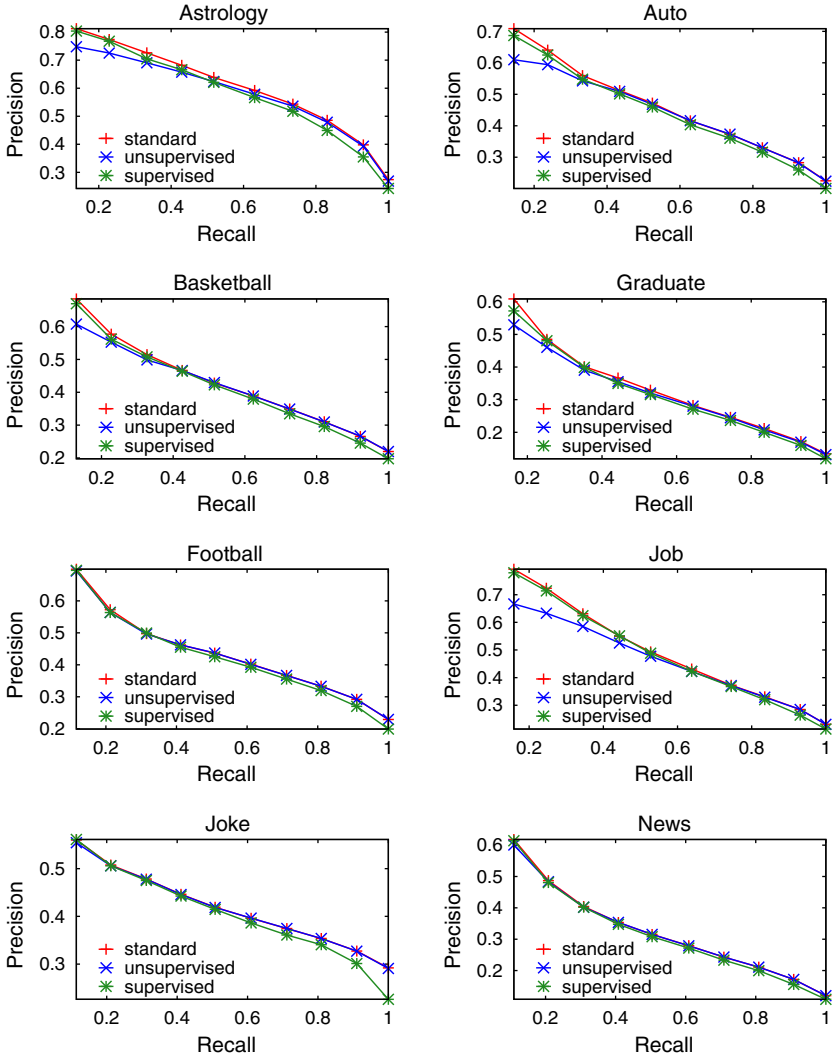

Fig. 8. Recommendation performance before and after randomization in eight subcommunities.

against cluster attack. We can see from the results that randomization techniques can dramatically reduce the performance of cluster attack. The proposed supervised randomization method can decrease total precision and cluster precision of Cluster Attack by 66.6% and 54.8%, respectively, on average over the eight subcommunities. The unsupervised randomization method can decrease total precision and cluster precision of Cluster Attack by 63.2% and 52.6%, respectively, on average over the eight subcommunities, which are also worse than the supervised randomization method.

Overall, randomization techniques are effective in protecting item-based top-N recommender systems from aggregated knowledge attacks, and the proposed supervised randomization method achieves better protection than the unsupervised method in all scenarios.

6.5.2. Recommendation accuracy loss comparison

Fig. 8 shows the recommendation accuracies of item-based top-N recommendation before and after randomization in eight subcommunities. The predefined recommendation accuracy loss threshold η is set to 5% in the supervised randomization. Results show that recommendation precision losses after the supervised randomization are less than 5% in all eight subcommunities (4.89% on average). This means that the supervised randomization can indeed keep the recommendation accuracy loss within the predefined threshold (5%). We can also see that the two randomization techniques achieve similar recommendation accuracy. However, precisions of the supervised randomization are higher than that of the unsupervised randomization when recalls are low (less than 0.4), and lower when recalls are high. This is because items

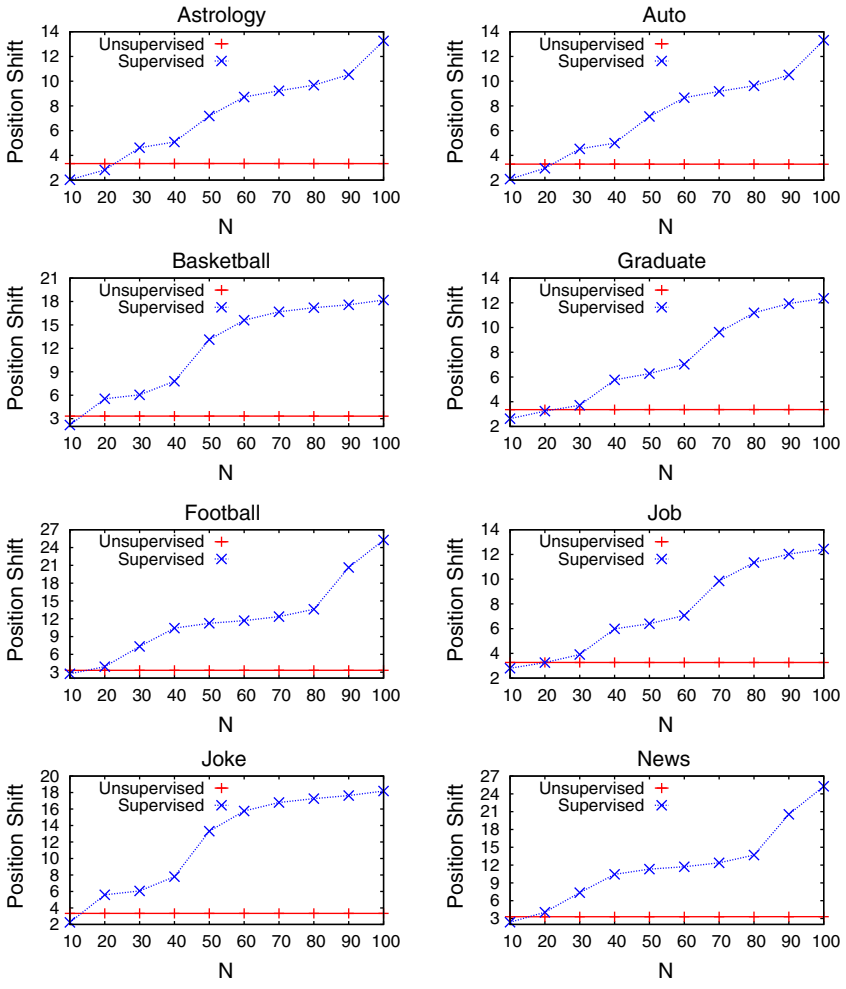

Fig. 9. Average item position shifts in the recommendation lists after randomization in eight subcommunities.

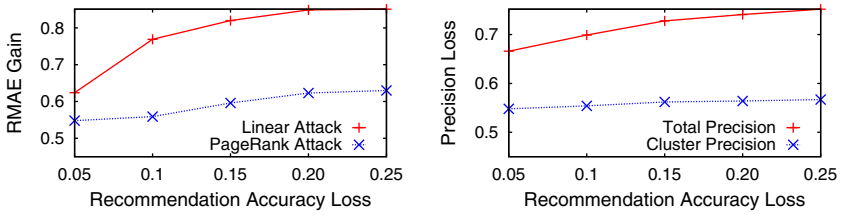

Fig. 10. Recommendation accuracy loss vs. aggregated information protection in the supervised randomization. (The left figure shows the tradeoff for item popularity ranking attacks, and the right figure shows the tradeoff for item clustering attack.)

are swapped in wider ranges in supervised randomization when more items are recommended, resulting in slightly worse accuracy. Typically, the number of recommended items is small in real world applications, e.g., around 50 in Amazon.com and around 40 in Youtube.com. Therefore, the supervised randomization method, which achieves better recommendation accuracy and better aggregated information protection, is a better choice for real world applications.

6.5.3. Item position shift after randomization

The positions of items are shifted in the recommendation list after randomization. The average *PositionShift* values can help us clearly know how the proposed supervised randomization method and the unsupervised method perform. Meanwhile, the results will help us better understand why the proposed supervised randomization method achieves higher recommendation precision when recalls are small and achieves lower precision when recalls are large as presented above.

As shown in Fig. 9, the unsupervised method achieves the same *Position Shift* values for all scenarios, which is because the unsupervised method adopts the Fisher-Yates shuffle equally on all sublists of the recommendation lists. However, the proposed supervised method achieves relatively smaller *PositionShift* when N is small (10–20) and achieves larger *PositionShift* when N is large. Smaller position shifts (around 2–3) will have smaller impacts on recommendation accuracy, thus the supervised method has higher recommendation accuracy when recall is small. Similarly, larger position shifts (around 3–25) will have larger impacts on recommendation accuracy, thus the supervised method achieves lower recommendation accuracy when recall is large. But, since the items ranked in lower positions are less likely to be liked by users, even large position shifts (10 or 20) will not affect too much the recommendation accuracy (as shown in Fig. 8).

6.5.4. Tradeoff between recommendation accuracy loss and aggregated information protection

In the proposed supervised randomization method, tradeoff must be carefully weighed between recommendation accuracy and aggregated information protection, i.e., better aggregated information protection indicates higher recommendation accuracy loss. As shown in Fig. 10, both RMAE gains of item popularity ranking attack and precision losses of cluster attack after randomization increase as recommendation accuracy loss increases. This indicates that better aggregated information protection can be achieved by compromising recommendation accuracy. In addition, the study shows that no significant increase of RMAE gains of item popularity ranking attack and precision losses of cluster attack are obtained when recommendation accuracy losses increase from around 10% to 25%. Therefore, a recommendation loss around 5–10% will be a good tradeoff, which can achieve desirable aggregated information protection (with around 55–75% degradation on performance of aggregated knowledge attacks) without severe compromise in recommendation accuracy.

7. Conclusion

In today's mainstream online services, item-based top-N recommender systems have been widely adopted, which provide recommendation services to end users based on inherent knowledge regarding the patterns or relations among users and items, such as their popularity or similarity. Such aggregated knowledge is of great importance to online services, and need be properly protected. In this paper, four attack methods are proposed and evaluated to demonstrate the severity of aggregated information revelation of item-based top-N recommendation. To make recommender systems more resilient to these attacks, a supervised randomization technique is proposed for item-based top-N recommendation. Experimental results demonstrate that the proposed attack methods can estimate the item popularity ranking and item clustering with high accuracy. Meanwhile, the proposed supervised randomization method can reduce the attack performance significantly. Based on the evaluation using real world dataset, our study shows that item-based top-N recommender system design should follow two guidelines in order to protect against aggregated knowledge attacks: (1) reduce the number of items in each recommendation list, and a number less than 50 will be a good choice and (2) item recommendation lists should be randomized using the proposed method before sending to end users, which can keep recommendation accuracy loss within a predefined threshold while protecting recommender systems from aggregated knowledge attacks.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant Nos. 61233016, 61332008 and 61272533, the National Science Foundation under awards CNS-0910995 and CNS-1162614, and the Shanghai Science & Technology Committee Project under Grant Nos. 11JC1400800 and 13ZR1401900.

References

[1] Greg Linden, Brent Smith, Jeremy York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Comput. 7 (1) (2003) 76–80.
[2] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, Dasarathi Sampath, The YouTube video recommendation system, in: Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys '10, ACM, 2010, pp. 293–296.
[3] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, Shyam Rajaram, Google News personalization: scalable online collaborative filtering, in: Proceedings of the 16th International Conference on World Wide Web, WWW '07, ACM, 2007, pp. 271–280.
[4] Badrul Sarwar, George Karypis, Joseph Konstan, John Reidl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, WWW '01, ACM, 2001, pp. 285–295.
[5] Manos Papagelis, Dimitris Plexousakis, Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents, Eng. Appl. Artif. Intell. 18 (7) (2005) 781–789.
[6] Jonathan Herlocker, Joseph Konstan, Al Borchers, John Riedl, An algorithmic framework for performing collaborative filtering, in: SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1999, pp. 230–237.
[7] Seung-Taek Park, David Pennock, Omid Madani, Nathan Good, Dennis DeCoste, Naive filterbots for robust cold-start recommendations, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, ACM, 2006, pp. 699–705.
[8] Dongsheng Li, Qin Lv, Xing Xie, Li Shang, Huanhuan Xia, Tun Lu, Ning Gu, Interest-based real-time content recommendation in online social communities, Knowl.-Based Syst. 28 (2012) 1–12.
[9] Marko Balabanović, Yoav Shoham, Fab: content-based collaborative recommendation, Commun. ACM 40 (1997) 66–72.
[10] Gediminas Adomavicius, Alexander Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (6) (2005) 34–749 (June).
[11] Jesus Bobadilla, Fernando Ortega, Antonio Hernando, Javier Alcalá, Improving collaborative filtering recommender system results and performance using genetic algorithms, Knowl.-Based Syst. 24 (8) (2011) 1310–1316.
[12] Jin-Min Yang, Kin Fun Li, Da-Fang Zhang, Recommendation based on rational inferences in collaborative filtering, Knowl.-Based Syst. 22 (1) (2009) 105–114.
[13] Qusai Shambour, Jie Lu, A trust-semantic fusion-based recommendation approach for e-business applications, Decis. Support Syst. 54 (1) (2012) 768–780.
[14] Qusai Shambour, Jie Lu, A hybrid trust-enhanced collaborative filtering recommendation approach for personalized government-to-business e-services, Int. J. Intell. Syst. 26 (9) (2011) 814–843.
[15] Huseyin Polat, Wenliang Du, Privacy-preserving collaborative filtering using randomized perturbation techniques, in: Proceedings of the Third IEEE International Conference on Data Mining, ICDM '03, IEEE, 2003, pp. 625–628.

[16] Sheng Zhang, James Ford, Fillia Makedon, A privacy-preserving collaborative filtering scheme with two-way communication, in: Proceedings of the 7th ACM Conference on Electronic Commerce, EC '06, 2006, pp. 316–323.
[17] Reza Shokri, Pedram Pedarsani, George Theodorakopoulos, Jean-Pierre Hubaux, Preserving privacy in collaborative filtering through distributed aggregation of offline profiles, in: Proceedings of the 3rd ACM Conference on Recommender Systems, RecSys '09, ACM, 2009, pp. 157–164.
[18] Bamshad Mobasher, Robin Burke, Runa Bhaumik, Chad Williams, Effective attack models for shilling item-based collaborative filtering systems, in: Proceedings of the 2005 WebKDD Workshop, ACM, 2005.
[19] Bamshad Mobasher, Robin Burke, Runa Bhaumik, Chad Williams, Toward trustworthy recommender systems: an analysis of attack models and algorithm robustness, ACM Trans. Internet Technol. 7 (4) (2007).
[20] Michael O'Mahony, Neil Hurley, Guénolé Silvestre, Recommender systems: attack types and strategies, in: Proceedings of the 20th National Conference on Artificial Intelligence, AAAI'05, AAAI Press, 2005, pp. 334–339.
[21] Kenneth Bryan, Michael O'Mahony, Pádraig Cunningham, Unsupervised retrieval of attack profiles in collaborative recommender systems, in: Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys '08, ACM, 2008, pp. 155–162.
[22] Shyong Lam, John Riedl, Shilling recommender systems for fun and profit, in: Proceedings of the 13th International Conference on World Wide Web, WWW '04, ACM, 2004, pp. 393–402.
[23] Sergey Brin, Lawrence Page, The anatomy of a large-scale hypertextual Web search engine, Comput. Netw. ISDN Syst. 30 (1–7) (1998) 107–117.
[24] James MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, 1967, pp. 281–297.
[25] Friedrich Leisch, A toolbox for K-centroids cluster analysis, Comput. Stat. Data Anal. 51 (2) (2006) 526–544.
[26] Dongsheng Li, Qin Lv, Huanhuan Xia, Li Shang, Tun Lu, Ning Gu, Pistis: a privacy-preserving content recommender system for online social communities, in: Proceedings of 2011 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT '11, 2011, pp. 79–86.
[27] Pang-Ning Tan, Vipin Kumar, Discovery of web robot sessions based on their navigational patterns, Data Min. Knowl. Discov. 6 (1) (2002) 9–35.
[28] Jin Li, Andrew Heap, A review of comparative studies of spatial interpolation methods in environmental sciences: performance and impact factors, Ecol. Informat. 6 (3–4) (2011) 228–241.
[29] U.S. National Institute of Standards and Technology, Dictionary of Algorithms and Data Structures, 2005.
[30] Xuetao Guo, Jie Lu, Intelligent e-government services with personalized recommendation techniques, Int. J. Intell. Syst. 22 (5) (2007) 401–417.
[31] Jie Lu, Qusai Shambour, Yisi Xu, Qing Lin, Guangquan Zhang, BizSeeker: a hybrid semantic recommendation system for personalized government-to-business e-services, Internet Res. 20 (3) (2010) 342–365.
[32] Neil Hurley, Robustness of recommender systems, in: Proceedings of the Fifth ACM Conference on Recommender Systems (RecSys '11), 2011, pp. 9–10.
[33] Giseop Noh, Chong-kwon Kim, RobuRec: robust Sybil attack defense in online recommender systems, in: 2013 IEEE International Conference on Communications (ICC), 2013, pp. 2001–2005.
[34] Giseop Noh, Young-myoung Kang, Hayoung Oh, Chong-kwon Kim, Robust Sybil attack defense with information level in online recommender systems, Expert Syst. Appl. 41 (4) (2014) 781–1791.
[35] Dongyan Jia, Fuzhi Zhang, Sai Liu, A robust collaborative filtering recommendation algorithm based on multidimensional trust model, J. Softw. 8 (1) (2013) 1–18.
[36] Benjamin Van Roy, Xiang Yan, Manipulation robustness of collaborative filtering, Manage. Sci. 56 (11) (2010) 1911–1929.