Research Article

Random Walk Based Location Prediction in Wireless Sensor Networks

Zhaoyan Jin, Dianxi Shi, Quanyuan Wu, and Huining Yan

College of Computer, National University of Defense Technology, Changsha 410073, China

Correspondence should be addressed to Zhaoyan Jin; jinzhaoyan@163.com

Received 3 September 2013; Revised 5 November 2013; Accepted 19 November 2013

Academic Editor: Yong Jin

Copyright © 2013 Zhaoyan Jin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the development of wireless sensor network (WSN) technologies, WSNs have been applied in many areas. In all WSN technologies, localization is a crucial problem. Traditional localization approaches in WSNs mainly focus on calculating the current location of sensor nodes or mobile objects. In this paper, we study the problem of future location prediction in WSNs. We assume the location histories of mobile objects as a rating matrix and then use a random walk based social recommender algorithm to predict the future locations of mobile objects. Experiments show that the proposed algorithm has better prediction accuracy and can solve the rating matrix sparsity problem more effectively than related works.

1. Introduction

Recently, WSNs have been applied in many areas, such as environmental monitoring [1], target tracking [2], and intrusion detection [3]. However, in all of these areas, location is very important, for data without location in WSNs will be useless. Research of localization is a hot topic in WSN, and it includes two kinds, that is, localization of sensor nodes in the WSN itself and localization of mobile objects using a WSN. For localization of sensor nodes in the WSN itself, several anchor nodes with known location are selected, and locations of other sensor nodes can be calculated with predefined anchor nodes [4]. For localization of mobile objects using a WSN, all sensor nodes are assumed to be anchor nodes, and the location of mobile objects can be calculated with observed anchor nodes [5]. The most popular localization system of this kind is the global position system (GPS).

Given a WSN, where all sensor nodes are anchor nodes, the locations of mobile objects can be calculated with the WSN. Furthermore, if we save all location records of mobile objects, then we can predict future locations of mobile objects with the observed location histories and the WSN. In this paper, we study the problem of future location prediction in WSNs. Here, we assume a WSN as a connected network, where each sensor node has connections with all its neighbors. The reason is that if a mobile object visits location \( A \), it will probably visit \( A \)'s neighborhood. In a WSN, each node records the number of visits of all mobile objects to itself. Hence, we have a network of sensor nodes and a rating matrix of sensor nodes on all mobile objects. The purpose of this paper is to predict which mobile object will probably visit the desired locations or sensor nodes in the future.

2. Related Works

If we assume each sensor node as a user and each mobile object as an item or a commodity, then the number of visits from a mobile object to a sensor can be assumed to a \(|\text{User}| \times |\text{Item}| \) rate matrix and the sensor network becomes a \(|\text{User}| \times |\text{User}| \) graph, and thus the location prediction problem can be transformed into a social recommender problem. In this section, we review related works on recommender systems.

2.1. Traditional Recommender Systems.

As mentioned by Huang et al. [6], collaborative filtering (CF) method is one of the most commonly used and successfully deployed methods in recommender systems. The CF recommender method can be classified into memory based and model based [3, 4].

The memory-based CF method is heuristic, and it aims to find a neighborhood of similar users [7, 8] or items [9–11] for recommendation. The user-based method predicts ratings
based on ratings of neighborhood users, and the item-based method predicts ratings based on ratings of neighborhood items. Here, the similarity between users or items could be Pearson correlation coefficient [12], vector space similarity [13], cosine similarity [14], and so on.

In model-based CF method, one can build a model by the priori User × Item matrix and predict unknown User-Item ratings with the model [15, 16]. The clustering models divide the matrix into several smaller clusters and use a neighborhood cluster to predict unknown user-item ratings [17, 18]. If the user-item ratings belong to different categories, then classification models can be applied [19]. Here, the priori User × Item matrix, where each element belongs to a category, is assumed to be training data. With the training data, classification model can be used to predict unknown ratings.

Recently, due to its efficiency in dealing with large datasets, several matrix factorization methods [20–24] have been proposed. The matrix factorization methods factor the User×Item matrix into two or three low-rank matrices and employ these low-rank matrices to make further predictions. The purpose of matrix factorization is to minimize the sum-squared errors of all elements in the priori User × Item matrix. Given a n × m rating matrix R_{nm}, the Singular Value Decomposition (SVD) [20, 21] is to factor the rating matrix R_{nm} into three low-rank matrices, that is, R_{nm} = A_{nk} \lambda_{nk} B_{km}, where k << n, m. First, one can factor the rating matrix R_{nm}, such that R_{nm} = A_{nk} \lambda_{nk} B_{km}, where \lambda_{nk} is a positive semidefinite diagonal matrix and \lambda_{ij} ≥ 0 for all 0 ≤ i ≤ j ≤ \min(m, n), and then truncate A_{nk}, \lambda_{nk}, and B_{km} into A_{jk}, \lambda_{jk}, and B_{km}, respectively, while keeping the k biggest singular values. Another matrix factorization (MF) method [22–24] is factoring the rating matrix R_{nm}, such that R_{nm} = U_{nk} V_{km} (k << n, m).

2.2. Trust-Based Recommender Systems. Traditional recommender systems have been well studied and developed in both academia and industry, but they all assume that users are independent and identically distributed and ignore the relationships between users. However, in practice, recommendations from different users should have different weights. For example, a preference or advice from a close friend should be more trustworthy than that of a stranger. Based on this idea, many researchers have recently begun to analyze trust-based recommender systems [25–30]. Trust-based recommender systems use both the rating matrix and the network of users to predict unknown ratings.

TidalTrust [25] applies the social network to find neighbors and aggregates their ratings weighted by the trusts between the source user and his neighbor. The searching of neighbors is based on breadth first search, which finds the nearest raters in the social network, and the trust between the source user and a rater is aggregating the trust value between source user’s direct neighbors and the rater, weighted by the trust values between the source user and its direct neighbors. MoleTrust [26] is similar to TidalTrust, except that the computation of trust between the source user and a rater is based on direct neighbors of the raters.

Golbeck and Hendler [27] studied how trust information can be mined and integrated into social-rating systems. They defined trust as “trust in a person is a commitment to an action based on a belief that the future actions of that person will lead to a good outcome,” presented two algorithms for inferring trust relationships between individuals that are not directly connected in the network, and proposed a prototype email client—TrustMail, which could filter spam for an email system.

TrustWalker [28] is a combination of item-based CF and trust-based recommendation, and it considers both ratings of similar items by trustworthy friends and ratings of the exact item by further neighbors in the social network. The random walk model of TrustWalker is as follows. Starting from the source user u, at step k and at node v, if v has rated i, then it returns to the rating r_{ij}, with the probability \Phi_{ijk}, the random walk stops, randomly chooses an item j rated by v, and returns to r_{ij} and with the probability 1 − \Phi_{ijk}, the random walk goes to a direct neighbor of v. If \Phi_{ijk} = 0, TrustWalker is a pure trust-based recommender system; if \Phi_{ijk} = 1, the random walk never starts, and thus TrustWalker becomes a traditional item-based CF recommender.

The works in [25–28] are all heuristic or memory based. However, model-based recommender methods, such as SocialTrustEnsemble [29] and SocialMF [30], can also be used in social recommender systems. Both SocialTrustEnsemble and SocialMF apply the matrix factorization method, and they factor both the User × User matrix and the User × Item matrix into two low-rank matrices. While choosing the neighborhood users, although they both use direct neighbors of the source user, they have different weighting methods. SocialTrustEnsemble gives different weights to ratings from different neighborhood users, whereas SocialMF assumes that the weight of a rating recursively depends on the neighbor’s neighbors.

2.3. Other Types of Recommendation. Besides recommending items for users, the social recommender systems can also recommend users [31–33], communities [34, 35], and tags [36, 37], as well as all kinds of recommendations for a group of users [38–40]. For more details about social recommender systems, readers can refer to [5].

3. SocialSlopeOne Location Prediction Algorithm

3.1. Problem Definition. In this paper, we consider each sensor node as a user, each mobile object as an item, and the number of visits from a mobile object to a sensor as the rating of user (sensor node) on that item (mobile object), and then we have a [User] × [User] graph and a [User] × [Item] rating matrix.

Assuming a WSN with n sensor nodes and m mobile objects, the WSN network G = (U, E) can be represented as an n × n adjacency matrix M_G:

\[
m_{G,ij} = \begin{cases} 1, & \text{if } (i, j) \in E \\ 0, & \text{otherwise,} \end{cases}
\]
Table 1: Notations.

| Notation | Description |
|----------|-------------|
| 𝑛        | Number of sensor nodes |
| 𝑚        | Number of mobile objects |
| 𝐺        | WSN graph |
| 𝑀𝐺       | Adjacency matrix of 𝐺 |
| 𝑃𝐺       | Transition probability matrix of 𝐺 |
| 𝑅        | Rating matrix |
| 𝑟𝑖,𝑗    | The rating of user 𝑖 on item 𝑗 |
| ̂𝑟𝑖,𝑗    | The predicted rating of user 𝑖 on item 𝑗 |
| 𝑀𝑅       | Unweighted adjacency matrix of 𝑅 |
| 𝑃𝑅       | Transition probability matrix of 𝑅 |
| 𝑁𝑜𝑖      | Neighbor nodes of outgoing edges for 𝑖 in 𝑅 |
| 𝑁𝑖       | Neighbor nodes of ingoing edges for 𝑖 in 𝑅 |
| π𝑖        | Stable distribution vector starting from 𝑖 |
| 𝑠𝑖,𝑗     | Similarity between user nodes 𝑖 and 𝑗 by the random walk model |

or a transition probability matrix 𝑃𝐺,

\[ 𝑃𝐺𝑖,𝑗 = \frac{𝑚𝐺𝑖,𝑗}{|𝑁𝑜𝑖|}, \]  

(2)

where |𝑁𝑜𝑖| is the set of nodes linked by outgoing edges of node 𝑖 and |𝑁𝑜𝑖| = \( \sum_{𝑗=1}^{𝑚} m_{𝐺𝑖,𝑗} \) is the cardinality.

Moreover, the rating matrix 𝑅 is 𝑛 × 𝑚, and

\[ 𝑟𝑖,𝑗 = \begin{cases} 𝑟𝑖,𝑗, & \text{if user } 𝑖 \text{ has rated item } 𝑗 \\ \text{null}, & \text{otherwise.} \end{cases} \]  

(3)

If we consider 𝑅 as an unweighted graph, then the adjacency matrix of 𝑅 is 𝑀𝑅,

\[ 𝑚𝑅𝑖,𝑗 = \begin{cases} 1, & \text{if } 𝑟𝑖,𝑗 \neq \text{null} \\ 0, & \text{otherwise,} \end{cases} \]  

(4)

and the transition probability matrix of 𝑅 is 𝑃𝑅,

\[ 𝑃𝑅𝑖,𝑗 = \frac{𝑚𝑅𝑖,𝑗}{|No𝑖|}, \]  

(5)

where |No𝑖| is the same as (2).

Given a WSN with a network 𝐺 and a rating matrix 𝑅, the purpose of the paper is to predict unknown ratings ̂𝑟𝑖,𝑗 = 𝑓(𝑖, 𝑗) in 𝑅 for 𝑟𝑖,𝑗 = null, where 𝑓 is built with 𝐺 and 𝑅. Table 1 lists all notations used in the paper.

3.2. Overall Recommender System. While predicting an unknown rating of user 𝑖 on 𝑗, our overall recommender system works as follows.

(i) It constructs a social-rating graph with the WSN graph and the rating matrix. Here, we merge the common nodes of users in the two graphs. Figure 1 is an example of a social-rating graph, and Table 2 is its rating matrix, where 𝑢𝑖 (1 ≤ 𝑖 ≤ 5) are users and 𝑖 (1 ≤ 𝑖 ≤ 5) are items.

(ii) While predicting the rating ̂𝑟𝑖,𝑗 of user 𝑖 on 𝑗, we compute the random walk based similarity, defined in Section 3.3, between user 𝑖 and the other users.

(3) Once we get the similarities between 𝑖 and other users, we can weigh the ratings of each user based on its similarity to 𝑖. With the rating matrix, we design a social slope one algorithm to predict the rating ̂𝑟𝑖,𝑗, and this is described in Section 3.4.

3.3. Random Walk Based Similarity. While predicting the rating ̂𝑟𝑖,𝑗 of user 𝑖 on 𝑗, we compute the similarities between user 𝑖 and all other users, that is, 𝑈 − {𝑖}, with a random walk with restart model. The main idea of the model is as follows:

(i) starting from the source node 𝑖;

(ii) at each step 𝑘 and at user node 𝑢:

(a) with probability 𝜖, return to 𝑖;

(b) with probability 1 − 𝜖, continue the random walk:

(1) firstly, randomly walk to a direct neighbor user 𝑢′ of 𝑖 in 𝑆 𝐺;

(2) secondly, randomly walk to a direct neighbor item 𝑖′ of 𝑖 in 𝑅;

(3) thirdly, randomly walk to a direct neighbor user 𝑢′′ of 𝑖 in 𝑅.

Figure 1: An example of social-rating graph.
The fingerprint of a random walk is a chain of nodes, which is $\langle \text{user}, \langle \text{user, item, user}, \langle \text{user, item, user}, \ldots, \langle \text{user, item, user} \rangle$ After the $k$th steps, the random walk would stop at a user node. For example, $\langle u_1, \langle u_2, i_1, u_2 \rangle, \langle u_3, i_2, u_4 \rangle \rangle$ is a fingerprint on Figure I with $k = 2$. We use the stable distributions, which are the probabilities of stopping at all user nodes, to represent the similarities between the source user $i$ and others and call them random walk based similarities. For a source user $i$, the random walk based similarity vector $\pi_i$ is

$$\pi_i = (1 - e) P_k P_k^T P_k^T \pi_i + e \delta_i,$$  

(6)

where the superscript $T$ is the transpose of a matrix, and $\delta_i$ is a personalized vector defined as follows.

$$\delta_i(j) = \begin{cases} 
1, & i = j \\
0, & i \neq j .
\end{cases}$$  

(7)

3.4. The Social Slope One Algorithm. As the slope one algorithm is an effective CF algorithm in recommender systems, we apply the idea of slope one for predicting unknown ratings. The slope one algorithm tries to find slope one suggestions in the rating matrix and uses slope one to predict unknown ratings. However, in a social recommender system, the rating matrix is sparse, and the slope one algorithm cannot predict ratings for users who do not have any slope one suggestion.

In order to solve the sparsity of the rating matrix, we propose a social slope one algorithm. While predicting a rating for user $u$ on $i$, our proposed algorithm works as follows.

3.4.1. Predicting Ratings for New Users. In social recommender systems, a new user is a person that has not rated any item yet, that is, $\text{No}_u = \emptyset$. In this situation, we can take suggestions for the specified item from other users. However, the suggestions from different users should have different weights. Here, we use the average of ratings, which other users have rated for the specified item, weighted by their similarities to the active user,

$$\bar{r}_{ui} = \frac{\sum_{v \in \text{Ni}_i} s_{uv} \cdot r_{vj}^j}{s_{uv}},$$  

(8)

where $s_{uv}$ is the normalized similarity between $u$ and $v$,

$$s_{uv} = \frac{\pi_u(v)}{\sum_{w \in \text{Ni}_i} \pi_u(w)}.$$  

(9)

Taking Table 2, for example, $u_4$ is a new user and $\bar{r}_{6.4}$ is the average of $r_{6.4}$ and $r_{5.4}$ weighted by $s_{6.2}$ and $s_{6.5}$, respectively, that is, $\bar{r}_{6.4} = s_{6.2} \cdot r_{6.4} + s_{6.5} \cdot r_{5.4}.$

3.4.2. Predicting Ratings for Non-Cold-Start Users. While predicting the rating $\bar{r}_{ui}$ of user $u$ on $i$, if user $u$ is not a new user and the item $i$ is not a new item, then we try to construct

slope one suggestions from the rating matrix, so that the slope one algorithm can be used to predict unknown ratings.

If all items that $u$ has rated have been rated only by itself, that is, $\text{Ni}_i = \{u\}$ for all $j \in \text{No}_u$, then we can treat $\bar{r}_{ui}$ as a new user problem, and use (8) to predict the rating. Taking user $u_4$ and item $i_2$ in Table I, for example, $\text{No}_i = \{i_1\}$ and $\text{Ni}_i = \{u_1\}$, so $\bar{r}_{1.5} = r_{1.5} + s_{1.2} \cdot r_{5.2}$. Otherwise, if $\text{Ni}_i \neq \{u\}$ for at least one $j \in \text{No}_u$, then we can construct slope one suggestions from the rating matrix, so that the slope one algorithm can be used. For each item $j \in \text{No}_u$, which has been rated by others besides the user $u$, we construct slope one suggestions using $\text{Ni}_j$ and $\text{Ni}_i$. While constructing the rating differences between items $i$ and $j$, at each time, we select one user $p$ from $\text{Ni}_j$ and one user $q$ from $\text{Ni}_i$ and minimize the following distance metric:

$$\text{dis}(p, q) = \frac{|s_{up} - s_{uq}|}{s_{up} \cdot s_{uq}}.$$  

(10)

Equation (10) means that it is better to choose two ratings for items $i$ and $j$ from users similar to each other, and moreover, both of them are similar to the active user. If users $p$ and $q$ are the same user, then $\text{dis}(p, q) = 0$ is the smallest, and thus the slope one suggestion between $i$ and $j$ is $r_{ij} = r_{pj} - r_{qj}$ and the weight is $(s_{up} + s_{uq})/2 = s_{u.q}$. If users $p$ and $q$ are not the same user, then the slope one suggestion is $r_{ij} = r_{pj} - r_{qj}$, and the weight is $(s_{up} + s_{uq})/2$. For $\text{Ni}_i$ and $\text{Ni}_j$, we can construct $\min(|\text{Ni}_i|, |\text{Ni}_j|)$ slope one suggestions and add them together weighted by the normalized distance, so the predicted rating $\bar{r}_{ui}$ using the slope one algorithm is

$$\bar{r}_{ui} = \sum_{r=1}^{\min(|\text{Ni}_i|, |\text{Ni}_j|)} \frac{\text{dis}_r}{\sum_{s=1}^{\min(|\text{Ni}_i|, |\text{Ni}_j|)} \text{dis}_s} \cdot \text{diff}_r + r_{ui},$$  

(11)

where the difference $\text{diff}_r$ and its weight $\text{dis}_r (1 \leq r \leq \min(|\text{Ni}_i|, |\text{Ni}_j|))$ are calculated by Algorithm I.

Then, for all $j \in \text{No}_u$ such that $\text{Ni}_j \neq \{u\}$, we add the predicted rating $\bar{r}_{uj}$ together weighted by $\min(|\text{Ni}_i|, |\text{Ni}_j|)$, that is, $w_{uj} = \min(|\text{Ni}_i|, |\text{Ni}_j|)$, and then $\bar{r}_{uj}$ can be calculated by the following equation:

$$\bar{r}_{uj} = \sum_{j \in \text{No}_u \land \text{Ni}_j \neq \{u\}} \frac{w_{uj}}{\sum_{k \in \text{No}_u \land \text{Ni}_k \neq \{u\}} w_{uk}} \bar{r}_{uj},$$  

(12)

Input: $\text{Ni}_i$ and $\text{Ni}_j$
Output: $\langle \text{diff}_r, \text{dis}_r \rangle$ for $1 \leq r \leq \min(|\text{Ni}_i|, |\text{Ni}_j|)$
(1) For $r = 1$ to $\min(|\text{Ni}_i|, |\text{Ni}_j|)$
(2) Find $p \in \text{Ni}_i, q \in \text{Ni}_j$ such that $\min\{\text{dis}(p, q)\}$
(3) $\text{diff}_r = r_{pj} - r_{qj}, \text{dis}_r = \frac{1}{2}(s_{up} + s_{uq})$
(4) Let $\text{Ni}_i = \text{Ni}_i - \{p\}, \text{Ni}_j = \text{Ni}_j - \{q\}$
(5) End for;
(6) Return $\langle \text{diff}_r, \text{dis}_r \rangle$ for $1 \leq r \leq \min(|\text{Ni}_i|, |\text{Ni}_j|)$

Algorithm I: SocialDiff($\text{Ni}_i$, $\text{Ni}_j$).
In Table 2, if we want to predict the rating $\hat{r}_{7,3}$, then $N_{07} = \{t_2, t_3\}$, $N_{t_2} = \{u_1, u_4, u_5, u_6\}$, and $N_{t_3} = \{u_3, u_7\}$. If we assume that $s_{t_3} = 1$ and $\hat{r}_{7,3} = 4 + ((3 - 4) + (4 - 5))/3 = 3/8$, $w_{2,7,3} = 3$, $w_{3,7,3} = 4 + ((4 - 5)/1)/3 = 2/3$, and finally, we can have $\hat{r}_{7,3} = (3/(1 + 3)) - (8/3) + ((1/(1 + 3)) - 3 \times 2.75$.

### 3.5. Complexity Analysis

In the problem of location prediction, our approach includes two steps, that is, computing nodes’ similarities with the random walk model and computing recommender results with the social slope one algorithm.

**Theorem 1.** The computational complexity of the random walk based similarity, defined by formula (6), is $k(r^2 + 2mn)$, where $k$ is the number of iterations.

**Proof.** The computation of formula (6) is the process of iterations, and the number of iterations decides the precision of the results. Here, we assume the number of iterations is $k$. At each iteration, there are three matrix-vector multiply operations, and their computational complexities are $n \times m$, $m \times n$, and $n \times n$, respectively. So, the computational complexity of formula (6) is $k(r^2 + 2mn)$.

**Theorem 2.** The computational complexity of the social slope one algorithm is $(1/3) \cdot (r/n) \cdot (r/m)^4$, where $r$ is the number of nonnull elements in the rating matrix $R$.

**Proof.** We assume that the rating matrix $R$ contains $r$ nonnull elements, and then each item is rated by about $r/m$ users and each user rates about $r/n$ items. While constructing slope one suggestions from one item, the SocialDiff($r/m, r/m$) algorithm iterates $r/m$ times (in line 1). At each iteration, the algorithm needs $i^2 \cdot (1 \leq i \leq r/m)$ time (from line 2 to line 4), that is, $(r/m) \cdot ((r/m) + 1) \cdot (2(r/m) + 1)/6$. So, the computational complexity of the social slope one algorithm is $(1/3) \cdot (r/n) \cdot (r/m)^4$.

From Theorems 1 and 2, we can have the following corollary.

**Corollary 3.** The computational complexity of our random walk based social slope one algorithm is $k(n^2 + 2mn) + (1/3) \cdot (r/n) \cdot (r/m)^4$.

### 4. Experiments

In this section, we perform experiments on two datasets, report the experimental results, and compare the results with existing CF algorithms. As the acquisition of sensor network data is very difficult, we use two social network data instead. However, both of them have similar characteristics with WSN data.

#### 4.1. Datasets

The datasets in our experiments are Epinions [26] and Flixster [30], and both of them contain a social network and a rating matrix. The Flixster dataset is a social network service, where users can rate movies, the relationships between users are bilateral, and the rating values are 10 discrete numbers in the range [0.5, 5] with step size 0.5. The Epinions dataset is also a social network service containing a social network and a rating matrix. However, the relationships in the Epinions dataset are unilateral, and the rating values are integers from 1 to 5. Basic statistics of the two datasets are in Table 3.

#### 4.2. Metrics

In the experiments, we treat the social network as known data and make 90% of the rating data as the training set and the remaining 10% as the test data. Our evaluation metrics are RMSE (root mean square error), precision, and coverage.

The RMSE of a model prediction with respect to the estimated variable $r_{ui}$ is defined as the square root of the mean squared error:

$$\text{RMSE} = \sqrt{\frac{\sum_{(u,i) \in R_{test}} (r_{ui} - \hat{r}_{ui})^2}{|R_{test}|}}, \quad (13)$$

where $|R_{test}|$ is the cardinality of test set $R_{test}$.

Precision (also called positive predictive value) is the fraction of retrieved instances that are relevant:

$$\text{Precision} = \frac{\#hit}{k}, \quad (14)$$

where $k$ is the number of the highest rating items that we recommend to the active user and $\#hit$ is the number of recommended items which are real in the top-$k$ list. In our experiments, we choose $k = 5$, that is, Precision@5.

Coverage is the ratio of successful recommendations in the whole test set:

$$\text{Coverage} = \frac{\#success}{|R_{test}|}, \quad (15)$$

where $\#success$ is the number of test data which we can make a recommendation successfully.

#### 4.3. Baseline Algorithms

In our experiments, we compare the predicting results of different algorithms. Following is

| Dataset  | No. of users | No. of edges | No. of items | No. of ratings | No. of new users |
|----------|--------------|--------------|--------------|----------------|------------------|
| Epinions | 49290        | 487183       | 139738       | 664824         | 7739             |
| Flixster | 786936       | 7058819      | 48794        | 8196077        | 36360            |

**Table 3:** Statistics of datasets.
4.4. Results. In this subsection, we present the results of our experiments, first for all users and then for cold start nodes, and finally evaluate the impact of number of neighbors on the results of the proposed SocialSlopeOne algorithm.

4.4.1. Performance on All Users. We evaluate the performances (RMSE, precision, and coverage) of all algorithms on the whole test set, and the results are listed in Table 4. For better illustrating the results, we compare the results in Table 4 with Figures 2 and 3.

As shown in Figure 2, for the Flixster dataset, SocialSlopeOne has similar prediction error to SlopeOneCF, and both of them have lower prediction error than other algorithms; for the Epinions dataset, SocialSlopeOne has the lowest prediction error. In Figure 3, for precision, SocialMF is the best in the Flixster dataset, whereas SocialSlopeOne is the best in the Epinions dataset; for coverage, SocialSlopeOne is the best in both datasets. Moreover, SocialSlopeOne is the best for the sum of coverage and precision for all datasets. Hence, we can have that SocialSlopeOne has the best prediction accuracy and coverage in both datasets.

4.4.2. Performance on Cold Start Nodes. We also evaluate the performances (RMSE, precision, and coverage) of all algorithms on cold start nodes, and the result is listed in Table 5. In our experiments, we define cold start nodes as ratings for new items and new users. For better illustrating the results, we compare the results in Table 5 with Figures 4 and 5.
Table 5: Experimental results for cold start nodes.

| Algorithm     | RMSE Epinions | RMSE Flixster | Precision Epinions | Precision Flixster | Coverage Epinions | Coverage Flixster |
|---------------|---------------|---------------|--------------------|--------------------|-------------------|-------------------|
| Pearson CF    | 1.382         | 1.239         | 0.218              | 0.453              | 0.369             | 0.417             |
| Slope One CF  | null          | null          | null               | null               | null              | null              |
| Trust Walker  | 1.218         | 1.205         | 0.734              | 0.793              | 0.735             | 0.665             |
| Social MF     | 1.289         | 1.182         | 0.609              | 0.729              | 0.701             | 0.682             |
| Social Slope One | 1.125         | 1.026         | 0.807              | 0.795              | 0.897             | 0.793             |

As shown in Figure 4, Social Slope One has the lowest prediction error in both datasets for cold start nodes. In Figure 5, for precision, Social Slope One and Trust Walker have similar precision, and they are better than others; for coverage, Social Slope One is also the best in both datasets. Moreover, Social Slope One is the best for the sum of coverage and precision for all datasets. Hence, we can conclude that Social Slope One has the best prediction accuracy and coverage for cold start nodes in both datasets.

4.4.3. Impact of $k$ on the Results.
In the Social Slope One algorithm, we need to construct $\min(|N_i|, |N_j|)$ slope one suggestions and distances from $N_i$ and $N_j$. In large $N_i$ and $N_j$, constructing all slope one suggestions will take a long time. However, small number of these suggestions, especially those from the same user, can approximate the results effectively. We set $k$ as the top list with small distances defined in (10) and use the top-$k$ slope one suggestions to approximate the results.

This top-$k$ method cannot be applied in new items, so we observe the RMSE along with $k$ for new users and all nodes. From Figure 6, we can see that, for both all nodes and new nodes, the RMSEs decrease with the rising of $k$. For small $k$, RMSEs decrease quickly, but as the increasing of $k$, RMSEs decrease slower and slower, which means that we can use small $k$ to approximate the results for large $N_i$ and $N_j$ while taking suggestions from other users.

5. Conclusions
Traditional localization technologies focus on calculating the location of sensor nodes or mobile objects using WSNs. However, this paper studies the problem of future location prediction for mobile objects in WSNs. In addition to the WSN, we assume the location histories of all mobile objects as a rating matrix and transfer future location prediction problem to a social recommender problem. In social recommender systems, although current recommender algorithms could give recommender results, they have either low prediction accuracy or high computational complexity and cannot deal with sparse data efficiently. In this paper, we propose a random walk based similarity metric to find similar users and use slope one suggestions from similar users to construct recommender results. With our defined similarity metric, we can find more similar users, and this solves the matrix
sparsity problem. In addition, we propose the social slope one recommender method, which has higher prediction accuracy and lower computational complexity. Experiments show that the proposed algorithm has better prediction accuracy and can solve the rating matrix sparsity problem more effectively than related works.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National High Technology Research and Development Program ("863" Program) of China (no. 2012AA011201).

References

[1] C. Alippi, R. Camplani, C. Galperti, and M. Roveri, "A robust, adaptive, solar-powered WSN framework for aquatic environmental monitoring," IEEE Sensors Journal, vol. II, no. 1, pp. 45–55, 2011.

[2] S. Bhatti and J. Xu, “Survey of target tracking protocols using wireless sensor network,” in Proceedings of the 5th International Conference on Wireless and Mobile Communications (ICWMC '09), pp. 110–115, IEEE, August 2009.

[3] A. P. R. Da Silva, A. A. F. Loureiro, M. H. T. Martins, L. B. Ruiz, B. P. S. Rocha, and H. C. Wong, "Decentralized intrusion detection in wireless sensor networks," in Proceedings of the 1st ACM International Workshop on Quality of Service and Security in Wireless and Mobile Networks (Q2SWinet ’05), pp. 16–23, ACM, October 2005.

[4] G. Mao, B. Fidan, and B. D. O. Anderson, “Wireless sensor network localization techniques,” Computer Networks, vol. 51, no. 10, pp. 2529–2553, 2007.

[5] A. Pal, “Localization algorithms in wireless sensor networks: current approaches and future challenges,” Network Protocols and Algorithms, vol. 2, no. 1, pp. 45–73, 2010.

[6] Z. Huang, H. Chen, and D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," ACM Transactions on Information Systems, vol. 22, no. 1, pp. 116–142, 2004.

[7] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 43–52, 1998.

[8] R. Jin, J. Y. Chai, and L. Si, "An automatic weighting scheme for collaborative filtering," in Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 337–344, July 2004.

[9] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," ACM Transactions on Information Systems, vol. 22, no. 1, pp. 143–177, 2004.

[10] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, 2003.

[11] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proceedings of the 10th International Conference on World Wide Web, pp. 285–295, 2001.

[12] J. Benesty, J. Chen, and Y. Huang, "On the importance of the pearson correlation coefficient in noise reduction," IEEE Transactions on Audio, Speech and Language Processing, vol. 16, no. 4, pp. 757–765, 2008.

[13] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," Communications of the ACM, vol. 18, no. 11, pp. 613–620, 1975.

[14] H. V. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," in 10th Asian Conference on Computer Vision (ACCV ’10), pp. 709–720, Springer, 2010.

[15] N. N. Liu and Q. Yang, "EigenRank: a ranking-oriented approach to collaborative filtering," in Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (ACM SIGIR ’08), pp. 83–90, July 2008.

[16] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," Society for Industrial Mathematics, vol. 5, pp. 471–480, 2005.

[17] C.-F. Tsai and C. Hung, "Cluster ensembles in collaborative filtering recommendation," Applied Soft Computing Journal, vol. 12, no. 4, pp. 1417–1425, 2012.

[18] S. K. Shinde and U. Kulkarni, "Hybrid personalized recommender system using centering-bunching based clustering algorithm," Expert Systems with Applications, vol. 39, no. 1, pp. 1381–1387, 2012.

[19] K. Balog, H. Ramampiaro, N. Takhirov, and K. Norvåg, "Multi-step classification approaches to cumulative citation recommendation," in Proceedings of the 10th Conference on Open Research Areas in Information Retrieval, pp. 121–128, 2013.

[20] C. Cobos, O. Rodriguez, J. Rivera et al., "A hybrid system of pedagogical pattern recommendations based on singular value decomposition and variable data attributes," Information Processing & Management, vol. 49, no. 3, pp. 607–625, 2013.

[21] A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro, "A hybrid content-based and item-based collaborative filtering approach
to recommend TV programs enhanced with singular value decomposition,” *Information Sciences*, vol. 180, no. 22, pp. 4290–4311, 2010.

[22] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[23] N. Guan, D. Tao, Z. Luo, and B. Yuan, “NeNMF: an optimal gradient method for nonnegative matrix factorization,” *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 2882–2898, 2012.

[24] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[25] J. A. Golbeck, *Computing and applying trust in web-based social networks [Ph.D. thesis]*, University of Maryland at College Park, 2005.

[26] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proceedings of the 1st ACM Conference on Recommender Systems (RecSys ’07)*, pp. 17–24, October 2007.

[27] J. Golbeck and J. Hendler, "Inferring binary trust relationships in Web-based social networks," *ACM Transactions on Internet Technology*, vol. 6, no. 4, pp. 497–529, 2006.

[28] M. Jamali and M. Ester, "TrustWalker: a random walk model for combining trust-based and item-based recommendation," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’09)*, pp. 397–405, July 2009.

[29] H. Ma, I. King, and M. R. Lyu, "Learning to recommend with social trust ensemble," in *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’09)*, pp. 203–210, July 2009.

[30] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proceedings of the 4th ACM Recommender Systems Conference (RecSys ’10)*, pp. 135–142, September 2010.

[31] N. L. Muscanell and R. E. Guadagno, "Make new friends or keep the old: gender and personality differences in social networking use," *Computers in Human Behavior*, vol. 28, no. 1, pp. 107–112, 2012.

[32] E. M. Daly, W. Geyer, and D. R. Millen, "The network effects of recommending social connections," in *Proceedings of the 4th ACM Recommender Systems Conference (RecSys ’10)*, pp. 301–304, September 2010.

[33] J. Hannon, M. Bennett, and B. Smyth, "Recommendng Twitter users to follow using content and collaborative filtering approaches," in *Proceedings of the 4th ACM Recommender Systems Conference (RecSys ’10)*, pp. 199–206, September 2010.

[34] W. Chen, J. Chu, J. Luan et al., "Collaborative filtering for orkut communities: discovery of user latent behavior," in *Proceedings of the 18th International Conference on World Wide Web*, pp. 681–690, 2009.

[35] A. Akther, H. Kim, M. Rawashdeh, and A. El Saddik, *Applying Latent Semantic Analysis To Tag-Based Community Recommendations*, Advances in Artificial Intelligence, Springer, 2012.

[36] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM ’10)*, pp. 81–90, February 2010.

[37] S. Tuorob, L. C. Pouchard, and C. L. Giles, "Automatic tag recommendation for metadata annotation using probabilistic topic modeling," in *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries*, 2013.
Submit your manuscripts at
http://www.hindawi.com