Knowledge-Based Systems 81 (2015) 148–158

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A link prediction approach for item recommendation with complex number

Feng Xie [a,b,*], Zhen Chen [b,c], Jiaxing Shang [a], Xiaoping Feng [d], Jun Li [b,c]

[a] Department of Automation, Tsinghua University, Beijing 100084, China
[b] Research Institute of Information Technology, Tsinghua University, Beijing 100084, China
[c] Tsinghua National Lab for Information Science and Technology, Beijing 100084, China
[d] AppChina Web Search Group, Beijing 100190, China

ARTICLE INFO

Article history:
Received 29 May 2014
Received in revised form 23 December 2014
Accepted 11 February 2015
Available online 18 February 2015

Keywords:
Recommender systems
Link prediction
Complex numbers
Data sparsity
Collaborative filtering

ABSTRACT

Recommendation can be reduced to a sub-problem of link prediction, with specific nodes (users and items) and links (similar relations among users/items, and interactions between users and items). However, previous link prediction approaches must be modified to suit recommendation instances because they neglect to distinguish the fundamental relations *similar* vs. *dissimilar* and *like* vs. *dislike*. Here, we propose a novel and unified way to cope with this deficiency, modeling the relational dualities using complex numbers. Previous works can still be used in this representation. In experiments with the MovieLens dataset and the Android software website AppChina.com, the proposed Complex Representation-based Link Prediction method (CORLP) achieves significant performance in accuracy and coverage compared with state-of-the-art methods. In addition, the results reveal several new findings. First, performance is improved, when the user and item degrees are taken into account. Second, the item degree plays a more important role than the user degree in the final recommendation. Given its notable performance, we are preparing to use the method in a commercial setting, AppChina.com, for application recommendation.

1. Introduction

Information overload creates difficulties for users. Information filtering tools, such as search engines, can help find items of interest to users, but the requirement that users specify in advance what they are looking for is still challenging [1,2]. Fortunately, recommender systems, which attempt to predict interests by mining data on past user-item interactions, can be used to identify what users need [3,4]. Consequently, recommender systems provide users with items that they are not aware of or cannot access using traditional keyword searching approaches. Recommender systems have been successfully deployed in many application settings, e.g., book, video, music, and friend recommendations on Amazon, Youtube, Pandora, and Facebook, respectively. Most of these have a client–server architecture with a centralized control mode. Some work has paved the way for developing recommender systems for personal knowledge management in collaborative environments in a distributed mode, widely used in connection with traditional knowledge management methods and tools [5,6].

An efficient recommender system can help customers find what they want quickly, and thereby save their time, improving the customer experience [7], and promoting sales [8]. As the core of a recommender system, recommendation algorithms typically take user and item attributes and user-item interactions (such as explicit ratings, and implicit browsing, purchasing or clicking-through activities) as input to anticipate user interests [9]. One of the most popular and promising recommendation algorithms, collaborative filtering (CF) provides recommendations using only user-item interactions [10,11], which can be classified as user based [12] or item based [13], depending on whether the cluster of recommendations are derived by identifying similar users based on their overlapping interactions or on similar items based on the common users who ever have expressed interests in them [14–16]. Despite its success, CF still suffers from data sparsity [17,18], where sparse user-item interactions lead to invalid user or item clustering. Some variants have been proposed to alleviate this problem [19–21]. Furthermore, this approach involves the risk that

* Corresponding author at: Department of Automation, Tsinghua University, China. Tel.: +86 15801438286.
E-mail address: xiefeng10@gmail.com (F. Xie).

http://dx.doi.org/10.1016/j.knosys.2015.02.013
0950-7051/

increasing numbers of users will be exposed to a narrowing selection of popular items, while unpopular items that might be very relevant to users will be overlooked [22]. Several attractive solutions have been proposed to overcome these disadvantages. One is to explore the structures of user-item interaction graphs to improve recommendation performance [23–25]. More specifically, users and items are regarded as nodes in a bipartite graph, with their interactions represented as links. In this representation, the recommendation problem is converted to finding future links for each user node, and thus can be converted into a link prediction problem. Link prediction is a fundamental problem that attempts to estimate the likelihood of the existence of a link between two nodes based on observed links and node attributes [26,27]. In a typical link prediction scenario, the nodes are symmetric, and the question of which node is the subject or object is neglected. However, there are two types of nodes in a user-item graph, users and items. In addition, three types of links (user-user, user-item, and item-item) depending on different endpoint combinations coexist. We further define the type of links between two users or items as *similar* or *dissimilar* and that between users and items as *like* or *dislike*. In this classical setting, it is much more interesting to predict *like* or *dislike* links since we typically would not recommend users to users or items to items.

In this paper, we propose a novel and unified model based on complex-number representation to address this task. The *similar* or *dissimilar* links are weighted by real numbers, while the *like* or *dislike* ones are weighted by complex numbers. Since complex number j has the property that $j^2 = -1$, complex numbers provide a natural way to model the particularities of item recommendations, when the recommendation problem is being reduced to a link prediction problem. Consequently, previous link prediction algorithms can still be used conveniently without modification. We evaluate the validity and efficiency of this representation and demonstrate the performance of this recommendation approach on two real-world datasets. One of the datasets is collected from our commercial platform, where the proposed method will be implemented in the near future.

The rest of this paper is organized as follows. Section 2 provides a detailed description of the proposed algorithm. Section 3 describes experiments on two real-world datasets and discusses the experimental results. This is followed by a final section, which summarizes the findings and proposes future research directions.

2. Proposed algorithm

The method proposed in this paper is based on abstracting recommendation to a link prediction problem. Firstly, the subjects (or users) and objects (or items) in a recommender system are regarded as nodes in a graph, while the links of the graph are taken to represent the relations between different types of nodes, such as user-user or item-item similarities and user-item interactions. Then, interest prediction between a particular user and an item can be reduced to evaluating the likelihood of existence of a link between the nodes corresponding to them in the graph. Since previous link prediction methods work by taking just one type of nodes into account, we need to modify them before using them in a recommendation scenario. This can be addressed efficiently with the proposed method by introducing complex numbers into graph theory.

2.1. Basic notation

In the typical link prediction approach based recommendation scenario, input data are modelled as a directed graph $G = (V, E, \omega)$ where the set of nodes V consists of all users U and items I present in the system ($V = U \cup I$), E is the set of links that represent various relations among these nodes ($E = U \times U \cup U \times I \cup I \times I$), and ω contains all of the links' weights. Furthermore, any path is denoted by $(a_1, a_2, \ldots, a_{k+1})(a_i \in V$, where $i = 1,2,\ldots,k+1$ and k is the length of the path), a_1 and a_{k+1} are two endpoints, while $a_i (i = 2,3,\ldots,k)$ is the inner node, and there are k links along this path ($(a_i, a_{i+1}) \in E$, where $i = 1,2,\ldots,k$). When $k = 1$, the length of the path is equal to one, and it is reduced to a link with no inner nodes. In addition, we define $N_i(u)$ as the set of items that are rated by user u and $N_u(i)$ as the set of users who have expressed interest in item i. That is, $N_i(u) = \{i \,|(u,i) \in E, i \in I\}$ and $N_u(i) = \{u \,|(u,i) \in E, u \in U\}$. If two nodes are connected, this node-pair is always connected by two links, one in each direction. Then, the recommendation is reduced to predicting whether a link will exist between an item and a particular user in the graph. In this paper, we calculate an estimated score that expresses how relevant any item is to a particular user using the link prediction algorithm.

2.2. Triangle closing

There are two types of relations among nodes in a user-item bipartite graph. First, there is the similarity, $\omega_{similar}$, between two users or items, including both user-user and item-item links. Second, there is the preference, ω_{like} and $-\omega_{like}$, of the user of an item, including user-item links and item-user links, respectively, because of the need to recognize the asymmetry between user and item. That is, when there is a link with weight ω_{like} from user u to item i, there is always a reverse link with weight $-\omega_{like}$ from item i to user u and vice versa. Here, ω_{like} and $\omega_{similar}$ are normalized values just for the weights. The principle of triangle closing in this model can be illustrated as in Fig. 1.

The principle is twofold: users who have expressed the same interest in (perhaps many) common items might be similar (see Fig. 1a), similar users will have a similar interest in the same item (see Fig. 1b), and user similarity is transitive among users (see Fig. 1c). Analogously, items liked by (perhaps many) common users might be similar (see Fig. 1d), users tend to be interested in similar items (see Fig. 1e), and item similarity is also transitive among items (see Fig. 1f). These are the core ideas of CF from another perspective. Consequently, these rules can be expressed mathematically as follows:

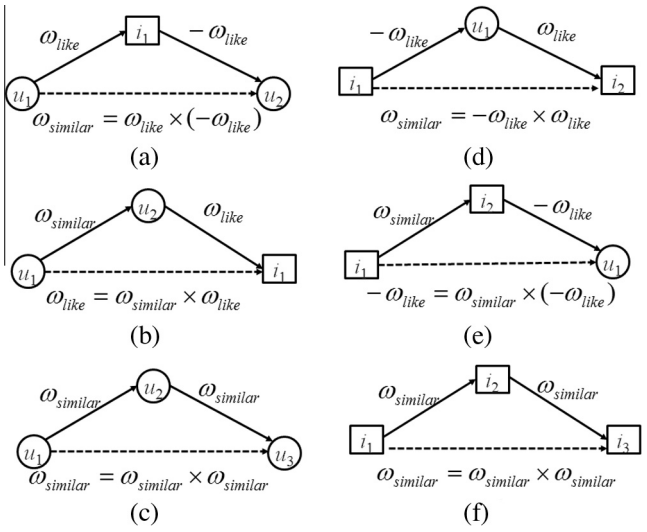


Fig. 1. The multiplication rules lead to triangle closing between the *like* and *similar* relations.

$$\omega_{similar} = -\omega_{like}^2 \quad (1)$$

$$\omega_{like} = \omega_{similar} \cdot \omega_{like} \quad (2)$$

$$\omega_{similar} = \omega_{similar}^2 \quad (3)$$

We thus must find two nonzero constants $\omega_{similar}$ and ω_{like} that solve this system of equations. The complex numbers provide a natural way to solve this problem, if we set $\omega_{like} = j$ and $\omega_{similar} = 1$ where j is the imaginary unit whose square is negative one. The above requirements then correspond to the identities $1 = -j^2, j = 1 \cdot j$ and $1 = 1^2$.

Analogous multiplication rules for *dislike* and *dissimilar* can then be derived by multiplying both sides by -1. For example, the case in which a user dislikes $(-j)$ an item dissimilar (-1) to one that he/she is interested in (j) can be interpreted as the equation: $-j = j \cdot (-1)$. In this representation, a link with real number weight has endpoints of the same type, two users or two items, and is always a real number. A greater value indicates more similar endpoints. In contrast, a link with an imaginary weight must be a user-item or item-user link, depending on the sign and interest. For example, if user u dislikes the item i, then there is a link with weight $-j$ from u to i and another link with weight j from i to u. In contrast to *similar* links, we can distinguish the *like* and *dislike* only when the direction of the link and the sign of its weight are known simultaneously. However, the weight's modulus value can represent the degree of *like* or *dislike*.

2.3. Extended triangle closing

With the basic triangle closings introduced in Section 2.2, the multiplication rules can be proved to be extendable to any path length, where the basic triangle closing is equivalent to the case of path length two. That is, the result of multiplying the weight of links along the path also depends only on the endpoints, independently of the inner nodes.

Lemma 2.1. *If the endpoints of a given path are two users or two items, then the result of multiplying the weights of all links along the path is a real number; if the endpoints are a user and an item, then the result is a complex number.*

Proof. We define k as the length of the path $(a_1, a_2, \ldots, a_{k+1})$ $(a_i \in V)$ and $p(k, (a_1, a_2, \ldots, a_{k+1}))$ as the result of multiplying the weights of all links along the path. That is, $p(k, (a_1, a_2, \ldots, a_{k+1})) = \prod_{i=1}^{k} \omega(a_i, a_{i+1})$, where $\omega(a_i, a_{i+1})$ $(\pm 1$ or $\pm j)$ is the weight of link (a_i, a_{i+1}). The proof using mathematical induction follows. First, if $k = 1$, then the path is reduced to the link $\omega(a_1, a_2)$, it thus satisfies the requirements

$$\begin{aligned} &p(1, (u_1, u_2)) = 1, p(1, (i_1, i_2)) = 1, \\ &p(1, (u, i)) = j, p(1, (i, u)) = -j \end{aligned} \quad (4)$$

Here the *dislike* and *dissimilar* cases are ignored, because they would not change the numeric type of the multiplication result. Then, we assume that the requirements are satisfied when $k = n$:

$$\begin{aligned} &p(n, (u_1, \ldots, u_2)) = 1, p(n, (i_1, \ldots, i_2)) = 1, \\ &p(n, (u_1, \ldots, i_1)) = j, p(n, (i_1, \ldots, u_1)) = -j \end{aligned} \quad (5)$$

Consequently, when $k = n + 1$, the paths with length $n + 1$ can be classified as: $(\underbrace{u_1, \ldots, u_2}_{n+1}, u_3)$, $(\underbrace{u_1, \ldots, u_2}_{n+1}, i_1)$, $(\underbrace{i_1, \ldots, i_2}_{n+1}, u_1)$, $(\underbrace{i_1, \ldots, i_2}_{n+1}, i_3)$, $(\underbrace{u_1, \ldots, i_1}_{n+1}, u_2)$, $(\underbrace{u_1, \ldots, i_1}_{n+1}, i_2)$, $(\underbrace{i_1, \ldots, u_1}_{n+1}, u_2)$, $(\underbrace{i_1, \ldots, u_1}_{n+1}, i_2)$. And then:

$$\begin{aligned} p(n+1, (u_1, \ldots, u_2)) &= p(n+1, (\underbrace{u_1, \ldots, u_3}_{n+1}, u_2)) \\ &= p(n, (u_1, \ldots, u_3)) \cdot p(1, (u_3, u_2)) \\ &= 1 \cdot 1 = 1 p(n+1, (u_1, \ldots, u_2)) \\ &= p(n+1, (\underbrace{u_1, \ldots, i_1}_{n+1}, u_2)) \\ &= p(n, (u_1, \ldots, i_1)) \cdot p(1, (i_1, u_2)) = j \cdot (-j) = 1 \end{aligned} \quad (6)$$

Namely, $p(n+1, (u_1, \ldots, u_2)) = 1$. Similarly, $p(n+1, (i_1, \ldots, i_2)) = 1$, $p(n+1, (u_1, \ldots, i_1)) = j$, and $p(n+1, (i_1, \ldots, u_1)) = -j$. That is, once the paths with length n satisfy the requirements, the paths with length $n + 1$ inherit that property. This proves the conclusion. Note that we ignore the subscripts for users and items only for descriptive convenience. □

Lemma 2.2. *If the graph is bipartite, there are only user-item links. When the length of any path k is even, its endpoints are two users or two items; when the endpoints are a user and an item, then the path length is odd.*

Proof. If we take only the user-item interactions into account, the user and item nodes will alternate along the path. Thus, when k is even, the path contains $k + 1$ (odd) nodes. Hence the endpoints are two users or two items. Analogously, the endpoints are different node types for paths of odd length. □

2.4. Adjacency matrix

Let $G = (V, E)$ be an unweighted, undirected network. The adjacency matrix of G is defined as $A \in \Re^{|V| \times |V|}$ given by:

$$A(x, y) = \begin{cases} 1 & \text{if } (x, y) \in E \\ 0 & \text{if } (x, y) \notin E \end{cases} \quad (7)$$

The adjacency matrix A is square and symmetric. Since the number of paths connecting two nodes can be derived by computing the powers of matrices in unweighted networks, the number of common neighbors between two nodes x and y $(x, y \in V)$ can be calculated as the square of the adjacency matrix: $N(x, y) = A^2(x, y)$, which implements the basic triangle closing and can be interpreted as the number of paths with length two between them. That number has an important property: greater the corresponding entry of the square of the adjacency matrix is, the closer the two nodes will be. Equivalently, we can extend the number of paths of any length k from node x to node y to be represented by the entry $A^k(x, y)$. Thus, the closeness of two nodes can be measured using the weighted sum of powers of the adjacency matrix A. An example of such a method for aggregating these results is the matrix exponential:

$$e^A = I + A + \frac{1}{2}A^2 + \cdots \quad (8)$$

The contributions of this function are twofold: it takes all paths between two nodes into account, since all powers of A are involved. In addition, short paths are given preference over long paths, as a result of the decreasing weights of the powers. Using the real numbers to represent the user-user and item-item relations and the complex numbers to describe the user-item interactions, the adjacency matrix A of the user-item graph G has the following property:

$$A(x,y)=\begin{cases}1 & \text{if } x \text{ similar } y\\ -1 & \text{if } x \text{ dissimilar } y\\ j & \text{if } x \text{ likes } y \text{ or } y \text{ dislikes } x\\ -j & \text{if } x \text{ dislikes } y \text{ or } y \text{ likes } x\\ 0 & \text{if } (x,y)\notin E\end{cases} \tag{9}$$

where $A(x,y)$ is the value in row x and column y of matrix A. Generally, matrix A can be denoted as $\begin{bmatrix}A_{UU} & A_{UI}\\ A_{IU} & A_{II}\end{bmatrix}$, where A_{UU}, A_{II} are the user and item similarity matrices, A_{UI}, A_{IU} are the user-item preference matrices, and A_{IU}=-A_{UI}^T. Obviously, the similarity matrices are real matrices, while the preference matrices are complex matrices. In this paper, we ignore the initial relations between users/items, hence G is a bipartite graph and the adjacency matrix A can be simplified to $\begin{bmatrix}0 & A_{UI}\\ -A_{UI}^T & 0\end{bmatrix}$. In accordance with the definition of the adjacency matrix A(see Eq. (9)), each entry in the preference matrix A_{UI} has only three candidate values, $j, -j$, and 0. Therefore, we can further convert A to $\begin{bmatrix}0 & jB\\ -jB^T & 0\end{bmatrix}$, where B is a real matrix.

Based on the path counting in the unweighted and undirected networks, the path counting for paths of length k can be derived similarly using A^k. If we take only the relations between users and items into account, Lemmas 2.1 and 2.2 can be further formulated mathematically as

$$A^k=\begin{cases}\begin{bmatrix}(BB^T)^n & 0\\ 0 & (B^TB)^n\end{bmatrix} & \text{where } k=2n\\ j\cdot\begin{bmatrix}0 & (BB^T)^nB\\ -(B^TB)^nB^T & 0\end{bmatrix} & \text{where } k=2n+1\end{cases} \tag{10}$$

Thus, any sum of the powers of the adjacency matrix A can be split into even and odd components. Here, we again take the matrix exponential as an example. This power sum can be applied to A, yielding

$$\begin{aligned}e^A&=I+A+\frac{1}{2}A^2+\frac{1}{6}A^3+\cdots\\&=\left(I+\frac{1}{2}A^2+\cdots\right)+\left(A+\frac{1}{6}A^3+\cdots\right)\\&=\left(I+\frac{1}{2}\begin{bmatrix}BB^T & 0\\ 0 & B^TB\end{bmatrix}+\cdots\right)\\&\quad+j\cdot\left(\begin{bmatrix}0 & B\\ -B^T & 0\end{bmatrix}+\frac{1}{6}\begin{bmatrix}0 & BB^TB\\ -B^TBB^T & 0\end{bmatrix}+\cdots\right)\end{aligned} \tag{11}$$

We can see that the even part of the power sum can be used to measure the similarities among users or items, while the odd part can be used to find items of interest to users. Therefore, only paths of odd length are suitable for recommendation. In the case of the matrix exponential, this is further reduced to the matrix hyperbolic sine:

$$sinh(A)=A+\frac{1}{6}A^3+\frac{1}{120}A^5+\cdots \tag{12}$$

2.5. Recommendation

Since the power sum of the adjacency matrix measures closeness among nodes, each entry of the top-right component expresses how relevant any item is to a particular user. Therefore, top-N recommendation can be generated by ranking items for each user with these estimated scores.

Given a system with three users $\{u_1,u_2,u_3\}$, six items $\{i_1,i_2,i_3,i_4,i_5,i_6\}$, and the user ratings of items $\{(u_1:i_1,5;i_2,1;i_3,4),(u_2:i_2,4;i_3,2;i_4,3),(u_3:i_1,5;i_3,5;i_4,4;i_6,1)\}$, if we further set three-star as the threshold for converting the discrete ratings to *dislike* or *like*, depending on whether the rating is less than the threshold, then the adjacency matrix with complex numbers can be formulated as (see Eq. (9))

$$A=\begin{array}{c}\\ u_1\\ u_2\\ u_3\\ i_1\\ i_2\\ i_3\\ i_4\\ i_5\\ i_6\end{array}\begin{array}{c}\begin{array}{ccccccccc}u_1 & u_2 & u_3 & i_1 & i_2 & i_3 & i_4 & i_5 & i_6\end{array}\\ \begin{pmatrix}0 & 0 & 0 & j & -j & j & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & j & -j & j & 0 & 0\\ 0 & 0 & 0 & j & 0 & j & 0 & j & -j\\ -j & 0 & -j & 0 & 0 & 0 & 0 & 0 & 0\\ j & -j & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ -j & j & -j & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & -j & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & -j & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & j & 0 & 0 & 0 & 0 & 0 & 0\end{pmatrix}\end{array} \tag{13}$$

where the user-user and item-item similarity matrices are zero matrices, since we ignore them. For simplicity, the third power (A^3) of the adjacency matrix A is computed with our proposed algorithm only in the prediction step, that is,

$$A^3=\begin{array}{c}\\ u_1\\ u_2\\ u_3\\ i_1\\ i_2\\ i_3\\ i_4\\ i_5\\ i_6\end{array}\begin{array}{c}\begin{array}{ccccccccc}u_1 & u_2 & u_3 & i_1 & i_2 & i_3 & i_4 & i_5 & i_6\end{array}\\ \begin{pmatrix}0 & 0 & 0 & 5j & -5j & 7j & -2j & 2j & -2j\\ 0 & 0 & 0 & -3j & 5j & -6j & 3j & -j & j\\ 0 & 0 & 0 & 6j & -3j & 7j & -j & 4j & -4j\\ -5j & 3j & -6j & 0 & 0 & 0 & 0 & 0 & 0\\ 5j & -5j & 3j & 0 & 0 & 0 & 0 & 0 & 0\\ -7j & 6j & -7j & 0 & 0 & 0 & 0 & 0 & 0\\ 2j & -3j & j & 0 & 0 & 0 & 0 & 0 & 0\\ -2j & j & -4j & 0 & 0 & 0 & 0 & 0 & 0\\ 2j & -j & 4j & 0 & 0 & 0 & 0 & 0 & 0\end{pmatrix}\end{array} \tag{14}$$

The top-right part of the matrix, shown in Eq. (14), can be used for recommendation. For each user (a line, for example, the predictions for user u_1 on the unrated items i_4, i_5, and i_6 are $-2j, 2j$, and $-2j$, respectively), we rank the predictions in descending order, with positive values indicating that the user will like the item and negative one representing a *dislike*. Thus, the items with positive and greater values will be recommended to a specific user, and these recommended items will not contain those that have been chosen by that user before. Then, i_5 will be recommended to u_1, since it receives a positive and greatest prediction value other than the already selected items. Intuitively, we can see that user u_1 and user u_3 both like items i_1 and i_3, as seen in the adjacency matrix A (Eq. (13)), hence u_1 will like i_5 (the prediction is a positive complex value of user u_1 on item i_5 in Eq. (14)), which u_3 has expressed interest in. Analogously, user u_1 and user u_2 expressed opposite interests in item i_2 and item i_3, hence u_1 would not be interested in what u_2 likes, i_4 (the prediction is a negative complex value of user u_1 on item i_4 in Eq. (14)), and u_2 also dislikes i_1 (the prediction is also a negative complex value of user u_2 on item i_1 in Eq. (14)) which is liked by u_1. From another perspective, this simply validates the idea of CF. Note that when two or more items receive the same prediction, the orders between them are determined either by their estimated scores of higher powers of the adjacency matrix or at random. Moreover, for a top-N recommender system, the first N items in the order other than the selected ones will be recommended to the particular user without concern for the signs of their predictions. Especially, returning to the example, if we want to provide a top-2 recommendation, the recommended items

for user u_1 will be $\{i_5, i_6\}$. Although u_1 provides i_6 and i_4 with the same prediction, only the former gets a ticket for recommendation, since its higher rank was generated between them randomly.

3. Experimental evaluation

To analyze the effectiveness of the proposed method, we conducted extensive experiments on two datasets comparing with several state-of-the-art methods using two quality metrics.

3.1. Datasets

The experiments were conducted on two real-world datasets, MovieLens[1] and AppChina[2]. The former is a publicly available movie rating dataset collected by GroupLens from the MovieLens website. It contains 100,000 ratings ranging from one to five by 943 users on 1682 movies. AppChina is an Android software installation tool that helps users download applications and games conveniently. Users' operations on applications, such as installation, updating, and deleting, were collected during the three-month period from May 1st, 2012 through July 31st, 2012. Then, the rating of a particular user on a specific application was modeled by aggregating this information. Detailed introductions regrading how to yield the ratings on a 1-to-5 scale are not shown here for lack of space. Finally, an available dataset with 99,295 ratings by 2395 users on 2486 applications was generated. Table 1 summarizes the statistical properties of these two datasets, where the sparsity level [28] is derived as

$$\#sparsity\ level = 1 - \frac{\#rating\ entries}{\#total\ entries} \tag{15}$$

Table 1
Properties of MovieLens and AppChina datasets.

Feature/dataset	MovieLens	AppChina
#Users	943	2395
#Items	1682	2486
#Total Ratings	100,000	99,295
#Average User Popularity	106	41
#Average Item Popularity	59	40
#Sparsity Level	93.7%	98.3%

3.2. Evaluation methodology

The evaluation methodology used in this paper is similar to the one in [29]. For each dataset, ratings are divided into two subsets, training and test. The test set contains only five-star ratings. Equivalently, only items relevant to the respective users are contained in the test set. The detailed procedure used to create the training and test sets can be described as follows. First, we randomly select 10% of the items rated by each user to form a temporary test set, with the temporary training set containing the remaining ratings. Then, the five-star ratings in the temporary test set are further filtered out for the final test set, and the rest of the ratings in the temporary test set are merged into the temporary training set for the final training set. In this case, the training set is used to obtain estimated ratings or recommendation scores for all user-item pairs.

In addition, rating conversion is required for the adjacency matrix generated in our proposed algorithm, in which the ratings in the training set are converted to $-j$ or j, depending on whether the rating is less than three. That is, if the rating is greater than or equal to three, it is replaced by j, indicating that the user expresses *like* for the item; analogously, when the rating is less than three, $-j$ is provided to represent *dislike*; moreover, if the (u, i) pair is not contained in the training set, the corresponding entry of the adjacency matrix receives zero (see Eq. (9)). With this dataset partitioning, computing the prediction error becomes less meaningful, so we care only about how many relevant items in the test set can be recommended to users. In addition, we focus on the overall ratio of items recommended to all users. Therefore, the metrics *hits rate* [29] and *coverage* [30,31] are used to measure the performance of the comparison methods. In the case of top-N recommendation, the overall *hits rate* and *coverage* are defined by averaging all the test cases:

$$hits\ rate(N) = \frac{\#hits}{|\ T\ |} \tag{16}$$

$$coverage(N) = \frac{|\ \cup_u\ recommend(N, u)\ |}{\#items} \tag{17}$$

For each pair (u, i) in the test set, if the item i is contained in the user u's top-N recommendation list, it will receive one *hit*. $\#hits$ is the overall number of occurrences of *hit*, and $|\ T\ |$ is the number of test pairs, with the result that *hits rate* can reasonably represent the ability to recommend relevant items to users. $recommend(N, u)$ is the item set recommended to user u; therefore, *coverage* corresponds to the percentage of items the system can recommend. *coverage* can usually be used to detect algorithms that, despite be highly accurate, recommend only a small number of items. To the best of our knowledge, a high *coverage* value is not only desirable, but is helpful for obtaining better trust accuracy metric results [32]. These two metrics share the property that greater values correspond to better algorithm performance.

3.3. Comparison methods

Some of the most popular and classical recommendation algorithms that are always taken as baseline methods are used here for comparison. Several of them perform very well in providing accurate prediction ratings. A detailed description follows:

Average: This method computes the average score for each item, and then recommends to users the items with greater scores. Every user receives the same recommendation list in this case, indicating that it is a non-personalized recommender algorithm.

Popular: Each item's popularity is measured by the number of users who have rated it. Greater item popularity means greater recommendation opportunities. This is also a non-personalized recommender algorithm, since it shows every user the same popular items. Unpopular items are omitted.

ItemBasedPear: This is a well-known item based CF approach [13], that calculates the similarity between two items using Pearson correlation measurement. Ease of use and interpretability have resulted in it being one of the most popular recommender methods.

SlopeOne: This is another family of algorithms used for CF [33]. Its simplicity makes it easy to implement and its prediction results (Root Mean Square Error, RMSE) are relatively accurate, while its storage and computation consumption are very high.

SVD++: This is the state-of-the-art method, based on the basic matrix factorization model [34]. The method yields reasonable prediction accuracy, but it is significantly more expensive computationally than other methods, owing to the requirement of iterative calculation.

CORLP: This is the proposed Complex Representation-based Link Prediction method, which uses complex numbers to represent the *like* and *dislike* relations between users and items. After the adjacency matrix is derived, its weighted power sum is computed.

[1] http://www.grouplens.org/
[2] http://www.appchina.com/

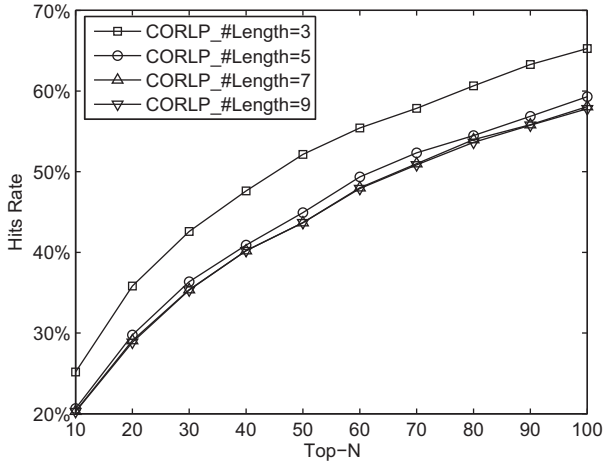
Then, we rank all items by their estimated scores in decreasing order for each user. The candidate recommendations for a particular user are the top-N items that have not yet been rated by that user.

4. Experimental results

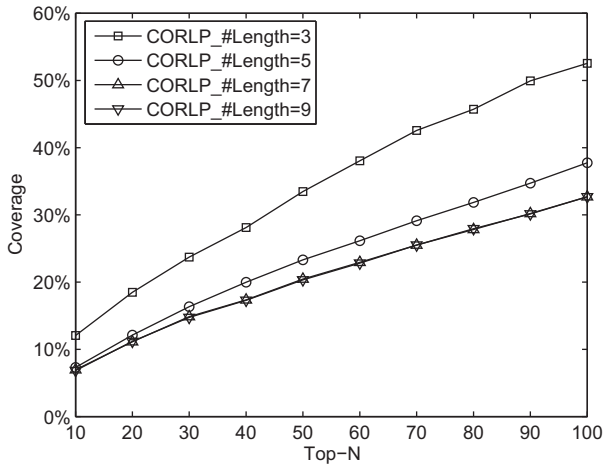
The experimental evaluation contains five parts. First, we study the effect of different path lengths used for recommendation by CORLP. Second, three numbers (3, 4, 5) for determining *like* and *dislike* relations are used to evaluate the effects of different thresholds. Third, several sequences of weighting factor are used to aggregate different path lengths, with the aims of showing that the sequences must be chosen carefully. Further, comparison results with competing methods are illustrated. Finally, we observe that the performance of CORLP is highly improved by taking user and item degrees into account.

4.1. Effect of the path length

Intuitively, the more paths there are between two nodes and the shorter those paths are, the stronger a relation the two nodes will have. Therefore, the first experiment is designed to test CORLP's performances with different path lengths for recommendation. For example, if the length is set to three, the predicted score of user u for item i is the value of the number of positive paths (the product of the links' weights along the path is positive) with length three minus the number of negative paths (the product of the links' weights along the path is negative) with length three from u to i regardless of other paths. Therefore, the more positive paths and the fewer negative paths there are from user u to item i, the greater opportunity i will have to be recommended to u. Note that the length must be odd and no less than three. Figs. 2 and 3 illustrate the *hits rate* and *coverage* comparison with lengths 3, 5, 7, and 9 on the MovieLens and AppChina datasets, respectively. The corresponding method is named with a combination of CORLP and the length, such as CORLP_#Length = 3. The results show that the *hits rate* and *coverage* increase as the number of recommended items increases. Moreover, CORLP_#Length = 3 greatly outperforms the other methods, with the performance decreasing sharply as the path length increases, but the rate of decrease in speed tends to slow as the path length becomes large enough. Experimentally, when the path length is greater than nine, the performance remains almost unchanged. Furthermore, CORLP performs much better on the MovieLens dataset than on the AppChina dataset, because the latter is much sparser. However, it still shows

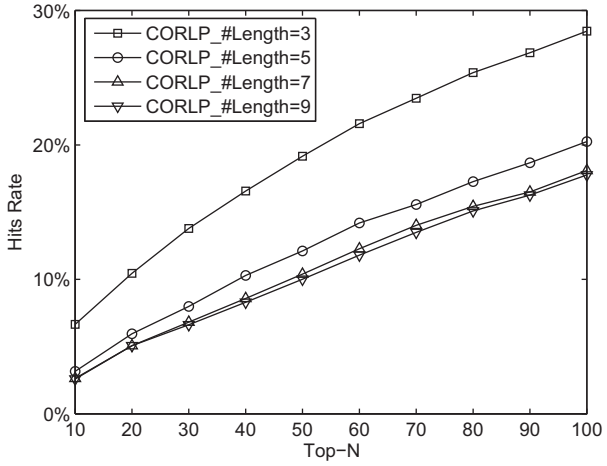


(a) *Hits Rate* Comparison on MovieLens

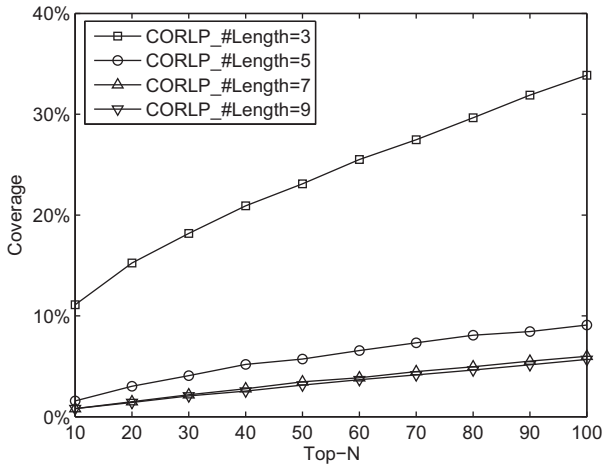


(b) *Coverage* Comparison on MovieLens

Fig. 2. The *hits rate* and *coverage* comparison of CORLP with different path lengths for recommendation, as top-N increases from 10 to 100 on MovieLens.



(a) *Hits Rate* Comparison on AppChina



(b) *Coverage* Comparison on AppChina

Fig. 3. The *hits rate* and *coverage* comparison of CORLP with different path lengths for recommendation, as top-N increases from 10 to 100 on AppChina.

attractive performance with length three for recommendation on the AppChina dataset.

As a similar consequence, the results with top-60 recommendation are shown separately for a clearer comparison. Fig. 4 illustrates these results of CORLP with lengths 3, 5, 7 and 9, respectively. It clearly shows the mentioned conclusions.

4.2. Effect of the threshold

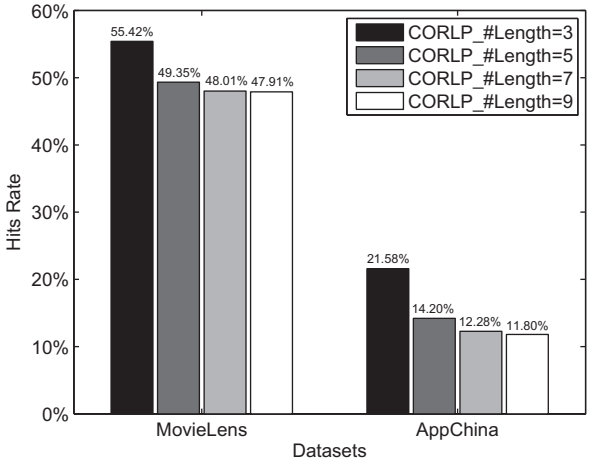
In our proposed methods, a rating is classified as *like* or *dislike* link, depending on whether it is greater than a given threshold. Because different thresholds generate different ratios between the numbers of *like* and *dislike*, graphs will differ, consequently influencing CORLP's performance. Therefore, we measure CORLP's performance by setting the threshold to be 3, 4, and 5, respectively. Correspondingly, the methods are denoted by combinations of CORLP and thresholds, such as CORLP_#Threshold = 3. If there is no special declaration, CORLP is the proposed method that uses only length three for recommendation. The results on the MovieLens dataset are shown in Fig. 5. CORLP_#Threshold = 4 achieves a slightly better *hits rate* and much higher *coverage* than CORLP_#Threshold = 3. Although CORLP_#Threshold = 5 achieves greatest *coverage*, its extremely low *hits rate* deprives its *coverage* of meaning. Therefore, CORLP performs best on the MovieLens dataset, when the threshold is set to four. Actually, the threshold does not change the number of links in the graph, but it influences the ratio of *like* and *dislike* links. A better threshold for CORLP would be one that can balance *like* and *dislike* links to some extent. Experimentally, we conclude that CORLP performs better when the threshold is set to the median of all ratings in the training set. The median is four and five for MovieLens and AppChina, respectively. Because the conclusions for the AppChina dataset are similar, those experimental results are not shown here.

4.3. Effect of the aggregating sequence

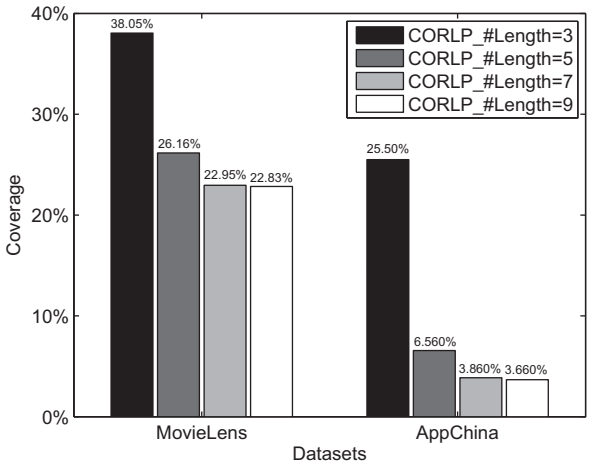
Although CORLP achieves outstanding performance only with length three for recommendation, it is worthwhile to try to aggregate separate results with different path lengths to generate a global recommendation. A general aggregation method can be formulated simply as

$$f(B) = \sum_{n=1}^{+\infty} a_n \cdot (BB^T)^n B \tag{18}$$

where B is the user-item preference matrix, for which the value of entry (u, i) is 1, −1, or 0, depending on whether user u likes, dislikes, or expresses no interest in item i, and $\{a_n\}$ is a decreasing sequence

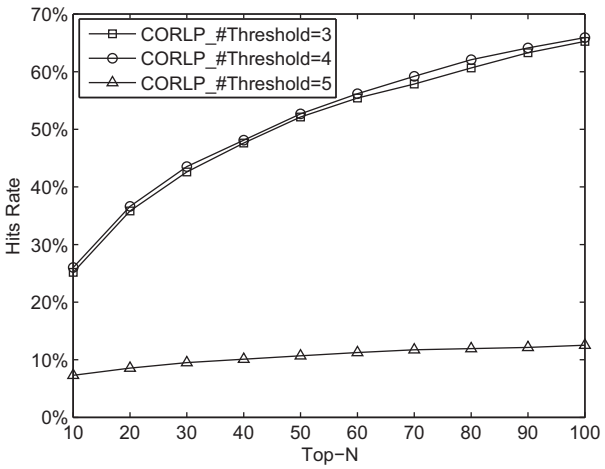


(a) *Hits Rate* Comparison on Two Datasets

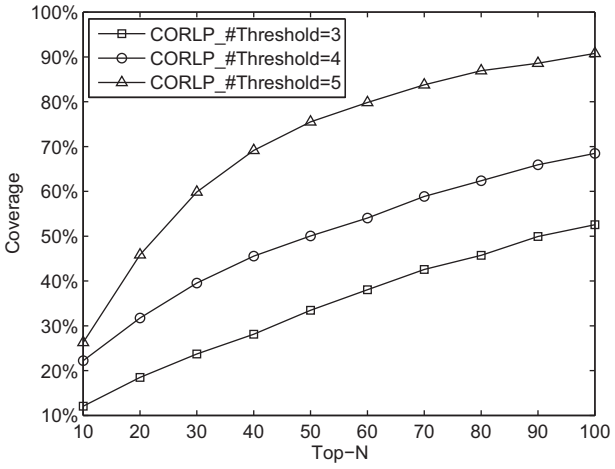


(b) *Coverage* Comparison on Two Datasets

Fig. 4. The *hits rate* and *coverage* comparison of CORLP with different path lengths for recommendation setting top-N to 60 on both datasets.



(a) *Hits Rate* Comparison on MovieLens



(b) *Coverage* Comparison on MovieLens

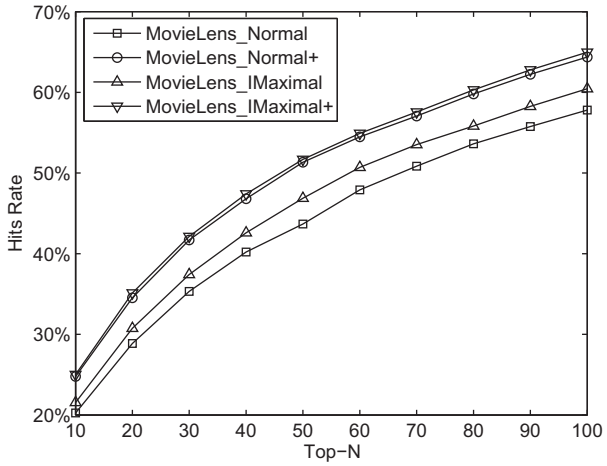
Fig. 5. The *hits rate* and *coverage* comparison of CORLP with different thresholds, as top-N increases from 10 to 100 on MovieLens.

Table 2
Average and maximal values of matrices with different path lengths.

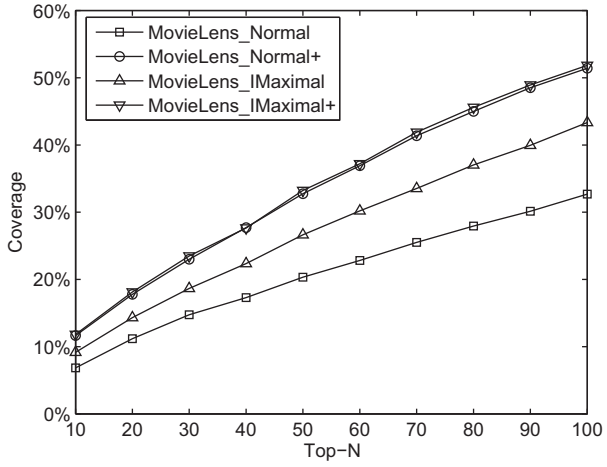
Scale/length	#Length = 3	#Length = 5	#Length = 7	#Length = 9
MovieLens_Average	6.91×10^2	1.17×10^7	1.99×10^{11}	3.37×10^{15}
MovieLens_Maximal	3.24×10^4	5.77×10^8	9.81×10^{12}	1.66×10^{17}
AppChina_Average	3.70×10^1	9.58×10^4	2.50×10^8	6.54×10^{11}
AppChina_Maximal	1.58×10^3	4.39×10^6	1.17×10^{10}	3.07×10^{13}

Table 3
Four aggregating sequences for each dataset.

Sequence/length	#Length = 3	#Length = 5	#Length = 7	#Length = 9
MovieLens_Normal	10^{-3}	10^{-5}	10^{-7}	10^{-9}
MovieLens_Normal+	10^{-3}	10^{-8}	10^{-13}	10^{-18}
MovieLens_IMaximal	$\frac{1}{3.24} \times 10^{-4}$	$\frac{1}{5.77} \times 10^{-8}$	$\frac{1}{9.81} \times 10^{-12}$	$\frac{1}{1.66} \times 10^{-17}$
MovieLens_IMaximal+	$\frac{1}{3.24} \times 10^{-4}$	$\frac{1}{5.77} \times 10^{-9}$	$\frac{1}{9.81} \times 10^{-14}$	$\frac{1}{1.66} \times 10^{-20}$
AppChina_Normal	10^{-3}	10^{-5}	10^{-7}	10^{-9}
AppChina_Normal+	10^{-3}	10^{-7}	10^{-11}	10^{-15}
AppChina_IMaximal	$\frac{1}{1.58} \times 10^{-3}$	$\frac{1}{4.39} \times 10^{-6}$	$\frac{1}{1.17} \times 10^{-10}$	$\frac{1}{3.07} \times 10^{-13}$
AppChina_IMaximal+	$\frac{1}{1.58} \times 10^{-3}$	$\frac{1}{4.39} \times 10^{-7}$	$\frac{1}{1.17} \times 10^{-12}$	$\frac{1}{3.07} \times 10^{-16}$



(a) *Hits Rate* Comparison on MovieLens



(b) *Coverage* Comparison on MovieLens

Fig. 6. The *hits rate* and *coverage* comparison of CORLP with different aggregating sequences on MovieLens.

of weighting factors guaranteeing that the estimated scores with shorter path lengths can contribute more to the final predictions. Because performance changes slightly, when the length is greater than nine, the lengths 3, 5, 7, and 9 are taken into account only for aggregation. First, it is necessary to analyze the scale of the values in matrices with different path lengths. Their average and maximal values are shown in Table 2. We can see that the values increase rapidly. Especially, the maximal values of matrices with different path lengths grow exponentially, and the approximate rate for the MovieLens dataset is 1.7×10^4, while it is 2.7×10^3 for the AppChina dataset, since the latter is much sparser than the former.

Given these observations, we designed four comparative experiments using different sequences of weighting factor for each dataset. Table 3 shows the details. MovieLens_Normal is the geometric series for CORLP on the MovieLens dataset whose decreasing ratio is 10^2, while MovieLens_Normal+ is an improved series with a much greater decreasing ratio of 10^5. The only difference is that the latter's decreasing ratio is much greater than the increasing ratio of the maximal values, while the inverse holds for the former. Moreover, it is intuitive that normalization is required before aggregating. Therefore, we use the inverse of maximal values, MovieLens_IMaximal, as a sequence. However, the sequence of MovieLens_IMaximal does not distinguish the degrees of importances of different path lengths. Therefore, an improved series MovieLens_IMaximal+ is generated by multiplying MovieLens_IMaximal to a geometric series $(10^0, 10^{-1}, 10^{-2}, 10^{-3})$, guaranteeing that shorter paths contribute more to the final results. Equivalently, AppChina_Normal, AppChina_Normal+, AppChina_IMaximal, and AppChina_IMaximal+ are defined similarly. Figs. 6 and 7 show their results for top-10 to top-100 recommendation on the MovieLens and AppChina datasets, respectively.

It is intuitive that *hits rate* and *coverage* increase when more items are recommended to users, and the experimental results confirm this expectation. Figs. 6 and 7 also show the same result that the proposed methods perform better on the MovieLens dataset than on the AppChina dataset, as derived above. Moreover, the methods with improved aggregating sequences outperform those with geometric series on both datasets. To explain this phenomenon, taking the MovieLens dataset as an example, if we multiply the geometric series $(10^{-3}, 10^{-5}, 10^{-7}, 10^{-9})$ by the vector of

the average values $(6.91 \times 10^2, 1.17 \times 10^7, 1.99 \times 10^{11}, 3.37 \times 10^{15})$ shown in Table 2, we obtain $(6.91 \times 10^{-1}, 1.17 \times 10^2, 1.99 \times 10^4, 3.37 \times 10^6)$, which implies that the part with length nine will dominate the final estimated scores in this case of aggregation. Consequently, the performance tends to be similar to that of the case in which only length nine is taken into account. However, with MovieLens_Normal+, the result of multiplication is $(6.91 \times 10^{-1}, 1.17 \times 10^{-1}, 1.99 \times 10^{-2}, 3.37 \times 10^{-3})$, which guarantees that the results with shorter length will contribute more to the final predictions. Naturally, it will yield more accurate recommendations using the improved aggregating sequences. Mathematically, we define the average number of co-rated items of each user pairs as *Com_item*, with *Per_user* and *Per_item* being the average number of items rated per user and the average number of users who have rated each item, respectively. Then, the average value of odd powers $2n+1$ of the adjacency matrix can be estimated approximately as

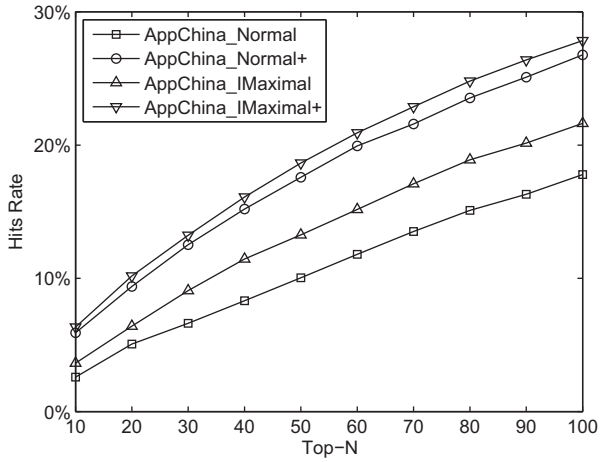
$$Avg(2n+1) \approx Com\_item \cdot (Per\_item \cdot Per\_user)^{n-1} \cdot Per\_item \quad (19)$$

Therefore, Eq. (19) can be a useful reference for generating aggregating sequences. The alternative method to generate aggregating sequences is firstly to normalize each power of the adjacency matrix using the inverse of maximal values so that the scale of each matrix is equivalently mapped to $[0, 1]$. Then a decreasing sequences is adopted to aggregate the normalized values. MovieLens_IMaximal+ and AppChina_IMaximal+ are two examples, which can achieve comparable or even better performance than the improved geometric series. Besides, MovieLens_IMaximal+ and AppChina_IMaximal+ also greatly outperform MovieLens_IMaximal and AppChina_IMaximal in *hits rate* and *coverage*, since the latter do not take account of the importance of path lengths.

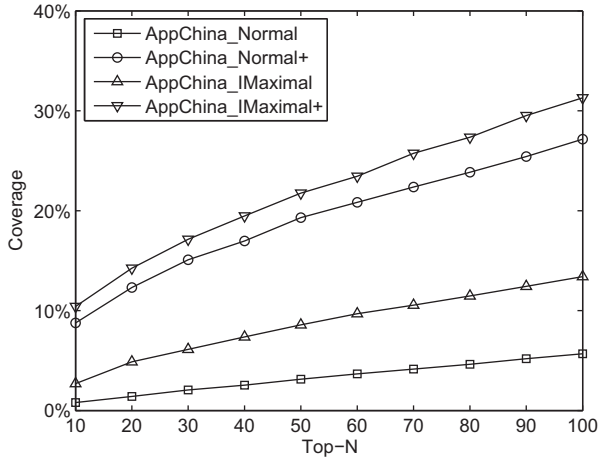
4.4. Comparison with other methods

Because CORLP can achieve high performance only with length three, the subsequent experiments for the proposed method take only length three into account. Fig. 8 shows the *hits rate* and *coverage* comparison with the recommender methods introduced in Section 3.3 on the MovieLens dataset.

CORLP outperforms other methods by *hits rate* and has relatively high *coverage*. ItemBasedPear suffers from low *hits rate*, while its *coverage* is very high, because it cannot make accurate recommendation when the dataset is sparse, resulting in a low *hits rate*. In this case, its *coverage* becomes less meaningful, since a high *coverage* value is desirable for comparable accuracy. Moreover, it is sur-

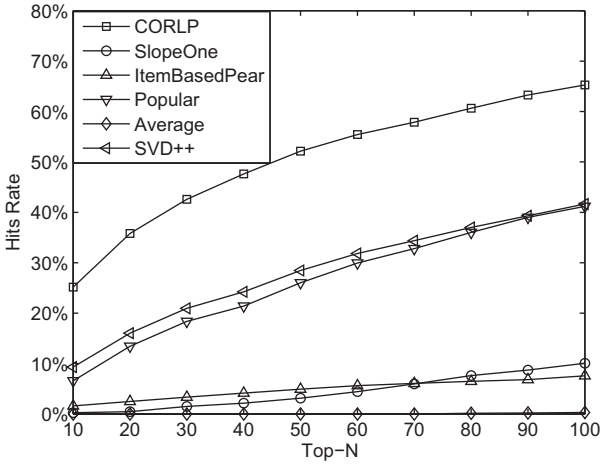


(a) *Hits Rate* Comparison on AppChina

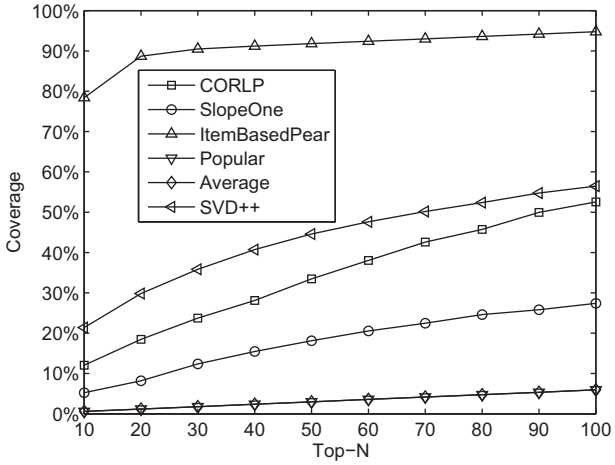


(b) *Coverage* Comparison on AppChina

Fig. 7. The *hits rate* and *coverage* comparison of CORLP with different aggregating sequences on AppChina.



(a) *Hits Rate* Comparison on MovieLens Dataset



(b) *Coverage* Comparison on MovieLens Dataset

Fig. 8. The *hits rate* and *coverage* comparison of the comparison methods on MovieLens. Average and Popular are two non-personalized methods that recommend the same items to users, so their *coverage* values are the same with two overlapping curves.

prising that the method that recommends only popular items to users obtained greater *hits rate* than ItemBasedPear and SlopeOne. This is a result of the fact that popular items will have high probabilities of appearance in the test set using the random partition method.

Fig. 9 illustrates similar results on the AppChina dataset. We can conclude that the proposed algorithm, CORLP, not only has higher recommendation accuracy, but also provides better *coverage*.

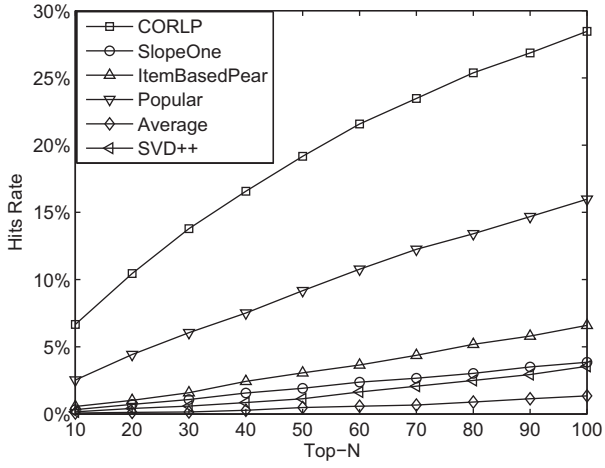
4.5. Effect of the user and item degrees

Using the basic link prediction approach based recommendation algorithm with complex numbers, we propose a series of improved methods. The basic method simply takes the number of paths and corresponding lengths between node pairs into account, assuming that more paths and shorter path lengths will result in closer node pairs. However, it does not take account of the importance of paths with the same length. Intuitively, the lower-degree (or popularity) nodes along the path would be expected to contribute more to the measurement of the closeness of two endpoints than those with higher-degree nodes. Here, we propose the CORLP_Item, CORLP_User and CORLP_User&Item methods. These improved algorithms differ slightly from the basic one, CORLP, in adjacency matrix modeling, while calculating of powers of the adjacency matrix and providing the final recommendation in the same manner. We define $|N_i(u)|$ and $|N_u(i)|$ as the degree of user u and item i, respectively. Then, each entry (u,i) of the adjacency matrix can be formulated as in Table 4.

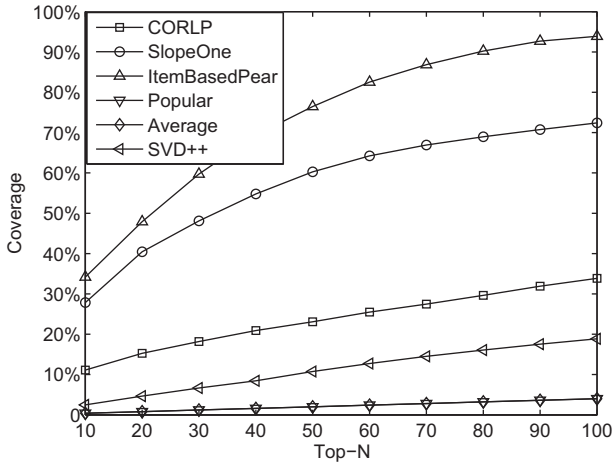
Fig. 10 shows the experimental results of four variants of CORLP by *hits rate* and *coverage*, as the number of items recommended for users ranges from 10 to 100 for the MovieLens dataset (similar results for the AppChina dataset are not provided). These algorithms share the property that the *hits rate* and *coverage* increase as the growth of the number of recommended items, and the improved ones achieve a higher *hits rate* than the basic CORLP method. We can also see that the recommended items will be more

Table 4
Entry comparison of complex number based algorithms.

Entry/method	CORLP	CORLP_Item	CORLP_User	CORLP_User&Item
like	j	$\frac{j}{\sqrt{\lvert N_u(i)\rvert}}$	$\frac{j}{\sqrt{\lvert N_i(u)\rvert}}$	$\frac{j}{\sqrt{\lvert N_u(i)\rvert\cdot\lvert N_i(u)\rvert}}$
dislike	$-j$	$-\frac{j}{\sqrt{\lvert N_u(i)\rvert}}$	$-\frac{j}{\sqrt{\lvert N_i(u)\rvert}}$	$-\frac{j}{\sqrt{\lvert N_u(i)\rvert\cdot\lvert N_i(u)\rvert}}$

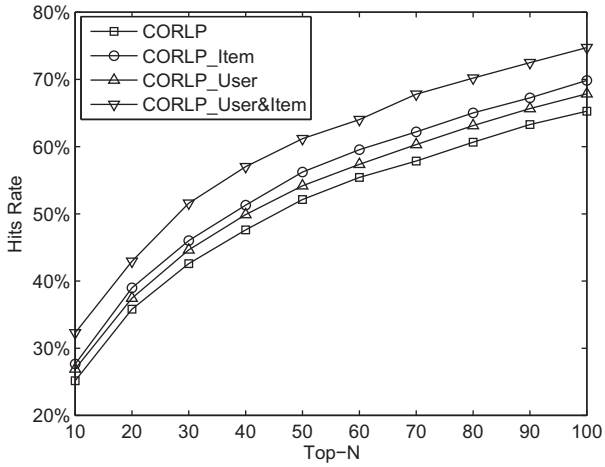


(a) *Hits Rate* Comparison on AppChina Dataset

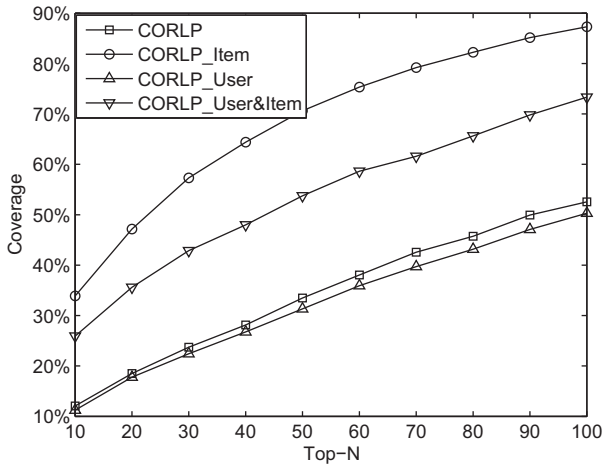


(b) *Coverage* Comparison on AppChina Dataset

Fig. 9. The *hits rate* and *coverage* comparison of the comparison methods on AppChina. Average and Popular are two non-personalized methods that recommend the same items to users, so their *coverage* values are the same with two overlapping curves.



(a) *Hits Rate* Comparison on MovieLens Dataset



(b) *Coverage* Comparison on MovieLens Dataset

Fig. 10. The *hits rate* and *coverage* comparison of the complex number based algorithms on MovieLens dataset.

relevant to users, when user and item degree are taken into account simultaneously, a result that derives from the fact that the CORLP_User&Item method outperforms other methods by *hits rate*.

Moreover, it can be determined that CORLP_Item obtains significantly better performance than CORLP_User by both *hits rate* and *coverage*, because user degree is less meaningful than item degree as a result of user subjectivity. In real-world cases, some users who have seen many movies might prefer not to provide comments. Hence considering their degrees to be small would be wrong. On the contrary, popular items will receive more ratings, while unpopular ones will receive fewer. Consequently, item degree appears to be more reliable. Note that even though the CORLP_User&Item method receives higher *hits rate* than CORLP_Item, its *coverage* appears to be much poorer.

5. Conclusions and future work

This paper proposed a link prediction based item recommendation method, CORLP. Using a complex representation for link weights in a graph, result in five contributions. First, CORLP can distinguish *similar* and *like* links efficiently, enabling the convenient reuse of previous link prediction approaches without any modifications. Second, experimental results indicated that CORLP outperforms state-of-the-art algorithms for the MovieLens and AppChina datasets for the *hits rate* and *coverage* metrics. Third, experimental results verified that recommendation is more efficient with shorter path lengths, and several methods were provided for aggregating the results from different path lengths to achieve better recommendations. Moreover, it was observed that the performance of CORLP is influenced substantially by the threshold set for classifying *like* and *dislike* links, and based on the experiment, we suggested choosing the median value as the most suitable one. Finally, to improve CORLP's performance, the user and item degrees were taken into account to recognize the importance of paths with the same length. The experimental results are extremely good, suggesting that item degree is more valuable than user degree.

The power of matrix calculation is time and space consumption as the number of users and items grows. It is essential to parallelize this method, to enable it to run on the AppChina.com website for application recommendation in the future. Moreover, the improved CORLP methods achieve relatively high performance, suggesting that it is worthwhile to continue developing futher factors to increase the power of CORLP.

Acknowledgment

The authors would like to thank Prof. Keqin Li of State University of New York at New Paltz for careful guidance regarding the structure and writing of the paper.

References

[1] U. Hanani, B. Shapira, P. Shoval, Information filtering: overview of issues, research and systems, User Model. User-Adap. Inter. 11 (2001) 203–259.
[2] N.J. Belkin, Helping people find what they don't know, Commun. ACM 43 (2000) 58–61.
[3] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. 17 (2005) 734–749.
[4] P. Resnick, H.R. Varian, Recommender systems, Commun. ACM 40 (1997) 56–58.
[5] L. Zhen, H.-T. Song, J.-T. He, Recommender systems for personal knowledge management in collaborative environments, Expert Syst. Appl. 39 (2012) 12536–12542.
[6] L. Zhen, Z. Jiang, H. Song, Distributed recommender for peer-to-peer knowledge sharing, Inf. Sci. 180 (2010) 3546–3561.
[7] Y. Jiang, J. Shang, Y. Liu, Maximizing customer satisfaction through an online recommendation system: a novel associative classification model, Decis. Support Syst. 48 (2010) 470–479.
[8] K.-W. Cheung, J.T. Kwok, M.H. Law, K.-C. Tsui, Mining customer product ratings for personalized marketing, Decis. Support Syst. 35 (2003) 231–243.
[9] M.J. Pazzani, A framework for collaborative, content-based and demographic filtering, Artif. Intell. Rev. 13 (1999) 393–408.
[10] D. Goldberg, D. Nichols, B.M. Oki, D. Terry, Using collaborative filtering to weave an information tapestry, Commun. ACM 35 (1992) 61–70.
[11] J.B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, in: The Adaptive Web, 2007, pp. 291–324.
[12] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, Grouplens: an open architecture for collaborative filtering of netnews, in: Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, 1994, pp. 175–186.
[13] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, 2001, pp. 285–295.
[14] J. Bobadilla, F. Ortega, A. Hernando, J. Alcalá, Improving collaborative filtering recommender system results and performance using genetic algorithms, Knowl.-Based Syst. 24 (2011) 1310–1316.
[15] J. Bobadilla, F. Serradilla, J. Bernal, A new collaborative filtering metric that improves the behavior of recommender systems, Knowl.-Based Syst. 23 (2010) 520–528.
[16] K. Choi, Y. Suh, A new similarity function for selecting neighbors for each target item in collaborative filtering, Knowl.-Based Syst. 37 (2013) 146–153.
[17] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowl.-Based Syst. 46 (2013) 109–132.
[18] H. Ma, T.C. Zhou, M.R. Lyu, I. King, Improving recommender systems by incorporating social contextual information, ACM Trans. Inf. Syst. (TOIS) 29 (2011) 9.
[19] F. Xie, M. Xu, Z. Chen, Rbra: a simple and efficient rating-based recommender algorithm to cope with sparsity in recommender systems, in: 2012 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2012, pp. 306–311.
[20] F. Xie, Z. Chen, H. Xu, X. Feng, Q. Hou, Tst: threshold based similarity transitivity method in collaborative filtering with cloud computing, Tsinghua Sci. Technol. 18 (2013) 318–327.
[21] F. Xie, Z. Chen, J. Shang, G.C. Fox, Grey forecast model for accurate recommendation in presence of data sparsity and correlation, Knowl.-Based Syst. 69 (2014) 179–190.
[22] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J.R. Wakeling, Y.-C. Zhang, Solving the apparent diversity-accuracy dilemma of recommender systems, Proc. Nat. Acad. Sci. 107 (2010) 4511–4515.
[23] Z. Huang, D. Zeng, H. Chen, A comparative study of recommendation algorithms in e-commerce applications, IEEE Intell. Syst. 22 (2007) 68–78.
[24] T. Zhou, J. Ren, M. Medo, Y.-C. Zhang, Bipartite network projection and personal recommendation, Phys. Rev. E 76 (2007) 046115.
[25] X. Li, H. Chen, Recommendation as link prediction in bipartite graphs: a graph kernel-based machine learning approach, Decis. Support Syst. 54 (2013) 880–890.
[26] L. Getoor, C.P. Diehl, Link mining: a survey, ACM SIGKDD Explor. Newslett. 7 (2005) 3–12.
[27] D. Liben-Nowell, J. Kleinberg, The link-prediction problem for social networks, J. Am. Soc. Inf. Sci. Technol. 58 (2007) 1019–1031.
[28] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Analysis of recommendation algorithms for e-commerce, in: Proceedings of the 2nd ACM Conference on Electronic Commerce, 2000, pp. 158–167.
[29] P. Cremonesi, Y. Koren, R. Turrin, Performance of recommender algorithms on top-n recommendation tasks, in: Proceedings of the Fourth ACM Conference on Recommender Systems, 2010, pp. 39–46.
[30] F. Gedikli, D. Jannach, Recommendation based on rating frequencies, in: Proceedings of the Fourth ACM Conference on Recommender Systems, 2010.
[31] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. (TOIS) 22 (2004) 5–53.
[32] F. Cacheda, V. Carneiro, D. Fernández, V. Formoso, Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems, ACM Trans. Web (TWEB) 5 (2011) 2.
[33] D. Lemire, A. Maclachlan, Slope one predictors for online rating-based collaborative filtering, in: SDM, vol. 5, 2005, pp. 1–5.
[34] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37.