A Link Prediction Approach for Item Recommendation with Complex Number

Feng Xie*†, Zhen Chen†‡, Jiaxing Shang*, Xiaoping Feng§, Wenliang Huang⟆ and Jun Li†‡

*Department of Automation, Tsinghua University, Beijing, 100084, China
†Research Institute of Information Technology, Tsinghua University, Beijing, 100084, China
‡Tsinghua National Lab for Information Science and Technology, Beijing, 100084, China
⟆China Unicom Groups, No. 21 Financial Street, Xicheng District, Beijing, 100140, China
§AppChina Web Search Group, Beijing, 100190, China
xief10@mails.tsinghua.edu.cn, zhenchen@tsinghua.edu.cn

***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, the previous link prediction algorithms need to be modified to suit the recommendation cases since they do not consider the separation of these two fundamental relations: *similar* or *dissimilar* and *like* or *dislike*. In this paper, we propose a novel and unified way to solve this problem, which models the relation duality using complex number. Under this representation, the previous works can directly reuse. In experiments with the MovieLens dataset and the Android software website AppChina.com, the presented approach achieves significant performance improvement comparing with other popular recommendation algorithms both in accuracy and coverage. Besides, our results revealed some new findings. First, it is observed that the performance is improved when the user and item popularities are taken into account. Second, the item popularity plays a more important role than the user popularity does in final recommendation. Since its notable performance, we are working to apply it in a commercial setting, AppChina.com website, for application recommendation.**

Keywords*—Recommender Systems, Link Prediction, Complex Number, Data Sparsity.

I. Introduction

Information overload makes users difficult to get what they want. Whereas information filtering tools, such as search engines, can help find users' interests, it is still facing a big challenge of the requirement for users to specify in advance what they are looking for [1][2]. Fortunately, recommender systems [3][4] are in charge of this challenge to identify users' needs, which attempt to predict interests by mining data on past user-item interactions. Consequently, recommender systems provide users items that they are not aware of or cannot access by traditional keyword searching approaches. Nowadays, recommender systems have been successfully deployed in many application settings, e.g., Amazon's book recommendation, movie recommendation for Netflix, music recommendation for Pandora, and friend recommendation on Facebook.

An efficient recommender system can help customers quickly find what they want, and save their time, which will virtually improve customer experience [5], and promote sales [6]. As the core of recommender systems, recommendation algorithms usually take user attributes, item attributes and user-item interactions (explicit ratings or implicit browsing, purchasing or clicking-through activities, etc.) as input to anticipate user interests [7]. As one of the most popular and promising recommendation algorithm, collaborative filtering (CF) [8][9] only takes advantage of user-item interactions to make recommendations, which can be further classified as user based [10] methods or item based [11] ones depending on whether the neighborhoods are derived by identifying similar users based on their overlapping interactions or similar items based on the common users who ever have expressed interests to them [12][13][14]. Despite its success, collaborative filtering suffers from data sparsity problem [15][16], where sparse user-item interactions lead to invalid neighbor clustering. To alleviate this problem, some variants are proposed [17][18][19]. Furthermore, the risk of this approach is that more and more users will be exposed to a narrowing band of popular items, while the unpopular ones that might be very relevant to users will be overlooked [20]. In order to overcome these disadvantages, several attractive solutions are proposed. One is to explore the structure of user-item interaction graphs to improve recommendation performance [21][22][23]. More specifically, the users and items are considered as nodes in a bipartite graph, while their interactions are regarded as links. Under this representation, the recommendation problem is converted to finding future links for each user node, and thus can be reduced to a link prediction problem. The link prediction [24][25] 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. In a typical link prediction scenario, the nodes are symmetric and we do not care about which node is the subject or object. However, there are two types of nodes in a user-item graph: users and items. Moreover, three types of links (user-user, user-item, and item-item) depending on different endpoint combinations will coexist. We further define the type of links between two users or items as *similar* or *dissimilar*, while the type of links between users and items as *like* or *dislike*. In this classical setting, it is much more interesting to predict *like* or *dislike* links, as we typically would not recommend users to users or items to items.

In this paper, we propose a novel and unified model to address this task based on complex number. The *similar* or *dissimilar* links are weighted by real numbers, while the *like* or *dislike* ones are weighted by complex numbers. Due to the

978-1-4799-4143-8/14 $31.00 © 2014 IEEE
DOI 10.1109/WI-IAT.2014.35

property of complex number j with $j^2 = -1$, complex numbers provide a natural way to model the particularities of item recommendations when reducing the recommendation problem to a link prediction problem. Benefiting from this, previous link prediction algorithms can be directly deployed without any modifications. We examine the validity and efficiency of this representation and demonstrate the performance of this recommendation approach using two real-world datasets. One of the datasets is collected from our commercial platform where the proposed algorithm will be implemented in the near future.

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

II. Proposed Algorithm

The proposed algorithm in this paper is based on the magic abstraction of recommendation to a link prediction problem. Firstly, the subjects (or users) and objects (or items) in a recommender system are regarded as nodes in the graph, while the links of the graph are the relations between different types of nodes, such as user-user, item-item similarities and user-item interactions. Then the interest prediction between the particular user and some item can be reduced to evaluating the likelihood of existence of a link between them in the graph. As the previous link prediction works just take one type of nodes into account, we need to modify them before using them in the recommendation scenario. With the proposed method, this can be efficiently addressed by introducing complex number into graph theory.

A. Basic Notation

In the typical link prediction approach based recommendation scenario, the input data are modelled as a directed graph G=(V, E, ω) 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 links' weights. Furthermore, any path is notated by (a_1,a_2,. . .,a_{k+1})($a_i \in V$, where i=1,2,. . .,k+1 and k is the length of this path), a_1 and a_{k+1} are two endpoints while a_i(i=2,3,. . .,k) is the inner node, and there are k links along this path ((a_i,a_{i+1}) $\in E$, where i=1,2,. . .,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 $N_u(i)$ as the set of items that are rated by user u and $N_i(u)$ as the set of users who have expressed interest to item i, respectively. That is, $N_u(i)$={i |(u,i)$\in E$,$i \in I$} and $N_i(u)$={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 link prediction algorithm.

B. Triangle Closing

As we know, there are two types of relations among nodes in the user-item graph, one is the similarity between two users or two items and the other is the preference of the user on an item with the notations of $\omega_{similar}$ (user-user or item-item links) and ω_{like} (user-item links), -ω_{like} (item-user links), respectively, since it is necessary to distinguish 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 -ω_{like} from item i to user u and vice versa. Here, ω_{like} and $\omega_{similar}$ are normalized values and just the weights' notations. In this model, the principle of triangle closing can be illustrated in Fig. 1.

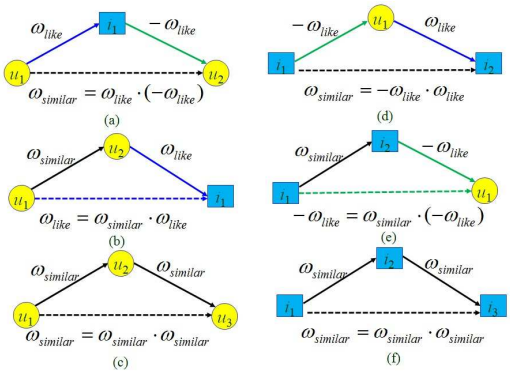


Fig. 1. The multiplication rules lead to the triangle closing between $like$ and $similar$ relationships.

The principle is two folds: users who have expressed the same interest to (maybe many) common items might be similar (see Fig. 1a); similar users will have similar interest to the same item (see Fig. 1b); and user similarity is transitive among users (see Fig. 1c). Analogously, items which are liked by (maybe many) common users might be similar (see Fig. 1d); user tends 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 collaborative filtering from another perspective. Consequently, these rules can be expressed mathematically in the following way:

$$\omega_{similar} = -\omega_{like}^2 \tag{1}$$

$$\omega_{like} = \omega_{similar} \cdot \omega_{like} \tag{2}$$

$$\omega_{similar} = \omega_{similar}^2 \tag{3}$$

We thus have to find two nonzero constants $\omega_{similar}$ and ω_{like} that solve these equations. It is observed that the complex numbers provide a natural way to solve these equations, when we set ω_{like}=j and $\omega_{similar}$=1 where j is the imaginary unit which squares to negative one. The above requirements then correspond to the identities 1=-j^2, j=1$\cdot j$ and 1=1^2.

Analogous multiplication rules with $dislike$ and $dissimilar$ can be then derived by multiplying both sides with -1. Under this representation, link with weight of real number has the same type of endpoints, two users or two items, and it is always a real number. The bigger the value is, the more similar two endpoints will be. Furthermore,

link with weight of imaginary number must be user-item or item-user link depending on the sign and interest. However, the weight's modulus value can represent the degree of $like$ or $dislike$.

C. Extended Triangle Closing

With the basic triangle closings introduced in Section II-B, it is provable to extend the multiplication rules to any length of paths, where the basic triangle closing is equivalent to the case of path length two. That is, the result of multiplication of links' weight along the path also just depends on the endpoints but independent of the inner nodes.

Lemma 2.1. Given a path, if its endpoints are two users or two items, the result of multiplication of all the links' weights along the path is a real number; while if the endpoints are the user and item, respectively, the result must be a complex number.

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; while the endpoints will be a user and an item, respectively, when path length k is odd.

D. Adjacency Matrix

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

$$A(x,y) = \begin{cases} 1 & if (x,y) \in E \\ 0 & if (x,y) \notin E \end{cases} \tag{4}$$

The adjacency matrix A is square and symmetric. As 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 formulated by 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. It has an important property: the bigger the entry of the square of the adjacency matrix is, the closer these 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 by the weighted sum of powers of the adjacency matrix A. An example of such a method to combine these results is the matrix exponential:

$$e^A = I + A + \frac{1}{2}A^2 + \dots \tag{5}$$

The contributions of this function are two folds: it takes all paths between two nodes into account since all powers of A are involved. Besides, short paths are given preference over long paths because of the decreasing weights of the powers. After using the real numbers to represent the user-user and item-item relations, and the complex numbers to describe the user-item interactions, respectively, the adjacency matrix A of the user-item graph G is such that:

$$A(x,y) = \begin{cases} 1 & if\ x\ similar\ y \\ -1 & if\ x\ dissimilar\ y \\ j & if\ x\ likes\ y\ or\ y\ dislikes\ x \\ -j & if\ x\ dislikes\ y\ or\ y\ likes\ x \\ 0 & if\ (x,y) \notin E \end{cases} \tag{6}$$

Where $A(x,y)$ is the value of row x and column y of matrix A. Generally, the 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, and 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 relations among users/items, then 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 adjacency matrix A(see Eq. (6)), each entry in the preference matrix A_{UI} only has three candidate values: $j, -j$ and 0. Therefore, we can further covert 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 weighted path counting for paths of length k can be similarly derived by A^k. If we only take the relations between users and items into account, LEMMA 2.1 and LEMMA 2.2 can be further mathematically formulated as:

$$A^k = \begin{cases} \begin{bmatrix} (BB^T)^n & 0 \\ 0 & (B^TB)^n \end{bmatrix} & where\ k = 2n \\ j \cdot \begin{bmatrix} 0 & (BB^T)^nB \\ -(B^TB)^nB^T & 0 \end{bmatrix} & where\ k = 2n+1 \end{cases} \tag{7}$$

Thus, any sum of the powers of the adjacency matrix A can be split into even and odd components, but only the odd components are useful for final recommendation. Therefore, the predictions can be generally applied to A giving:

$$P(A) = \alpha_1 \cdot A^3 + \alpha_2 \cdot A^5 + \alpha_3 \cdot A^7 + \alpha_4 \cdot A^9 + \dots \tag{8}$$

To guarantee the shorter paths contribute more to the predictions, $\{\alpha_1, \alpha_2, \alpha_3, \dots\}$ is a decreasingly weighting sequence.

E. Recommendation

As the power sum of the adjacency matrix measures the closeness among nodes, and 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.

III. Evaluation

In order to analyze the effectiveness of the proposed algorithm, we have conducted extensive experiments on two datasets using different algorithms and quality metrics.

A. Comparison Methods

Some of the most popular and classical recommendation algorithms are adopted for comparison, which are always taken as baseline methods. Several of them perform very well in the accuracy of rating prediction. The detailed introduction is as follows:

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

Item Popularity: Each item's popularity is measured by the number of users who have rated it. Larger item popularity means greater recommendation opportunities. Similarly, this is also a non-personalized recommender algorithm since it shows every user the same popular items. In this case, unpopular items are overlooked.

Item based CF: This is a well-known memory based collaborative filtering approach [11], which calculates the similarity between two items with Pearson correlation measurement. Easy to deploy and interpretability make it one of the most popular recommender methods.

Slope One: This is also a family of algorithms used for collaborative filtering [26]. Its simplicity makes it easy to implement and its prediction results (Root Mean Square Error, RMSE) are relatively accurate, while the storage and computation consumption are very high.

Matrix Factorization model (MF) based Recommendation: This is the state-of-the-art method, which is based on the basic matrix factorization model [27]. The method yields reasonable prediction accuracy but it is significantly more computationally expensive than other methods due to the requirement of iterative calculation.

Link Prediction Approach with Complex Number: It is the proposed algorithm, which is based on link prediction method with complex number representing the $like$ and $dislike$ relations between users and items. After deriving the adjacency matrix, the weighted power sum of the adjacency matrix is computed firstly. Then we rank all items by their estimated scores decreasingly for each user, and the candidate recommendations are top-N items which have not yet been rated by the particular user.

B. Datasets

The proposed algorithm and other comparison methods were conducted on two real-world datasets: MovieLens[1] and AppChina[2]. The former is a publicly available movie rating dataset, which were collected by GroupLens research from MovieLens website, which consists of 100,000 ratings range from 1 to 5 from 943 users on 1,682 movies. AppChina is an Android software installation tool, which makes users conveniently download applications and games. Users' operations to 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 certain application is modeled by aggregating this information. The detailed introduction about how to yield the ratings in 1-to-5 scale is not shown here for lack of space. Finally, an available dataset with 99,295 ratings from 2,395 users on 2,486 applications is generated. Table I summarizes the statistical properties of these two datasets, where the sparsity level [28] is derived as:

$$\sharp sparsity\ level = 1 - \frac{\sharp rating\ entries}{\sharp total\ entries} \tag{9}$$

TABLE I. PROPERTIES OF MOVIELENS AND APPCHINA DATASETS.

Feature/Dataset	Movielens	AppChina
♯Users	943	2,395
♯Items	1,682	2,486
♯Total Ratings	100,000	99,295
♯Average User Popularity	106	41
♯Average Item Popularity	59	40
♯Sparsity Level	93.7%	98.3%

C. Testing Methodology

The testing methodology adopted in this paper is similar to the one in [29]. For each dataset, ratings are split into two subsets: training set and test set. The test set contains only 5-stars ratings. Equivalently, only items relevant to the respective users are contained in the test set. The detailed procedure used to create the training set and the test set can be described as: First, we randomly select 10% of items rated by each user to form a temporary test set, while the temporary training set contains the remaining ratings; Then the 5-stars ratings in the temporary test set are further filtered out for the final test set, and the rest of 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.

Besides, the rating converting is needed for the adjacency matrix generating of our proposed algorithm, where the ratings in the training set are converted to -j or j depending on whether the rating is less than 3 or not. That is, if the rating is greater than or equal to 3, it is replaced by j, which means that the user expresses $like$ to the item; analogously, when the rating is less than 3, -j is given to represent the $dislike$; moreover, if the (u, i) pair isn't contained in the training set, the corresponding entry of the adjacency matrix gets zero (see Eq. (6)). With this dataset partitioning, computing the prediction error becomes less meaningful, so we only care about how many relevant items in the test set can be recommended to users. In addition, we also focus on the overall ratio of recommended items 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 the top-N recommendation, the overall $hits\ rate$ and $coverage$ are defined by averaging all test cases:

$$hits\ rate(N) = \frac{\sharp hits}{|T|} \tag{10}$$

$$coverage(N) = \frac{|\cup_u recommend(N, u)|}{\sharp items} \tag{11}$$

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 get one hit. $\sharp hits$ is the overall hit and $|T|$ is the number of test pairs, so $hits\ rate$ can reasonably represent the capability to recommend relevant items to users. $recommend(N, u)$ is

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

the item set recommended to user u, therefore, $coverage$ corresponds to the percentage of items the system is able to recommend. $coverage$ can be usually used to detect algorithms that, despite good accuracy, recommend only a small number of items. To the best of our knowledge, a high $coverage$ value is not only desirable, but it is helpful to better trust accuracy metric results [32]. These two metrics present the same property that the bigger the values are, the better the performance of the algorithm is.

D. Experimental Results

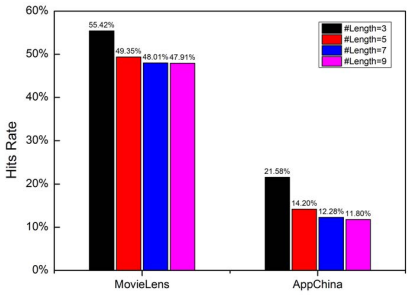
Intuitively, the more paths between two nodes and the shorter these paths are, the more strong relation these two nodes will have. Therefore, the first experiment was designed to test link prediction approach based recommender algorithms' performances with different path lengths for recommendation. For example, if the length is set to three, the predicted score of user u to item i is the value of the number of positive paths (the result of multiplication of all links' weights along the path is positive) with length three subtracting the number of negative paths (the result of multiplication of all links' weights along the path is negative) with length three from u to i regardless of other lengths of paths. Therefore, the more positive paths from user u to item i and the less negative paths between them, the greater opportunity i will get to be recommended to u. We should note that the length must be odd, and no less than three. As the similar consequence, the results with top-60 recommendation are only given. Fig. 2 illustrates these results of the proposed method with length 3, 5, 7 and 9, respectively.

It shows that the $hits\ rate$ and $coverage$ decrease as the path length increases on two datasets and the decreasing speed tends to be slow as the path length becomes larger. Experimentally, when the path length is greater than 9, the performance almost remains unchanged. Moreover, the proposed algorithm performs much better on MovieLens dataset than on AppChina dataset since the latter is much sparser. However, it still shows the attractive performance with length 3 for recommendation on AppChina dataset. At this point, it is worth trying to aggregate separate results with different path lengths to generate a global recommendation. A general aggregating method can be simply formulated as:

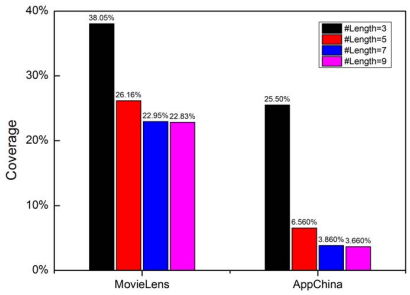
$$f(B) = \sum_{n=1}^{+\infty} a_n \cdot (BB^T)^n B \tag{12}$$

Where B is the user-item preference matrix, and the value of its entry (u, i) is 1, -1, or 0 depending on whether user u likes, dislikes or do not express interest to item i. $\{a_n\}$ is a decreasing sequence of weighting factor which guarantees that the estimated scores with short path length can contribute more to the final predictions. Because the performance changes a little when the length is greater than 9, the length 3, 5, 7 and 9 are only taken into account for aggregation. Here, we design two comparative experiments using different sequences of weighting factor for each dataset. Table II shows the details.

$MovieLens\_Normal$ is the geometric series of weighting factor for the MovieLens recommendation, while $MovieLens\_Improve$ is an improved one. The



(a)



(b)

Fig. 2. The $hits\ rate$ and $coverage$ comparison of the proposed algorithm with different lengths of paths for recommendation.

TABLE II. TWO SERIES OF WEIGHTING FACTOR FOR EACH DATASET.

Algorithm/Length	$\sharp Length = 3$	$\sharp Length = 5$	$\sharp Length = 7$	$\sharp Length = 9$
$Movielens\_Normal$	10^{-3}	10^{-5}	10^{-7}	10^{-9}
$Movielens\_Improve$	10^{-3}	10^{-9}	10^{-14}	10^{-19}
$AppChina\_Normal$	10^{-3}	10^{-5}	10^{-7}	10^{-9}
$AppChina\_Improve$	10^{-3}	10^{-7}	10^{-12}	10^{-16}

$AppChina\_Normal$ and $AppChina\_Improve$ are defined similarly. Fig. 3 shows their results of top-10 to top-100 recommendation, respectively.

It is intuitive that the $hits\ rate$ and $coverage$ increase when more items are recommended to users, and the experimental results just confirm this conclusion. Fig. 3 also shows the same result that the proposed algorithm performs better on MovieLens dataset than on AppChina dataset as derived above.

Moreover, the algorithms with improved series of weighting factor outperform the ones with geometric series of weighting factor on both datasets. In order to explain this phenomenon, we deeply observe the entries' values of different power of the adjacency matrix. The result of respective average value is given in Table III.

Obviously, the average value grows extremely fast as the increasing of path length. Take MovieLens dataset as an example, if we multiply the geometric series

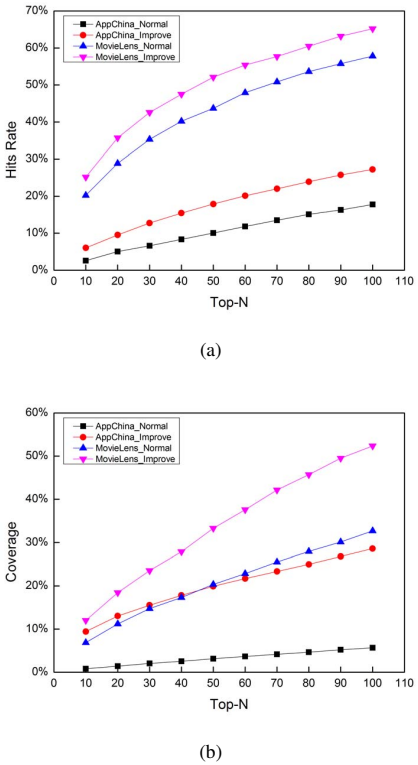


Fig. 3. The $hits\ rate$ and $coverage$ comparison of the proposed algorithm with different weighting factor series.

TABLE III. AVERAGE VALUES OF MATRICES WITH DIFFERENT PATH LENGTHS.

Dataset/Length	$\sharp Length = 3$	$\sharp Length = 5$	$\sharp Length = 7$	$\sharp Length = 9$
$Movielens$	6.91×10^2	1.71×10^7	1.99×10^{11}	3.37×10^{15}
$AppChina$	3.70×10^1	9.58×10^4	2.50×10^8	6.54×10^{11}

$(10^{-3}, 10^{-5}, 10^{-7}, 10^{-9})$ to the vector of the average value $(6.91\times10^2, 1.71\times10^7, 1.99\times10^{11}, 3.37\times10^{15})$ shown in Table III, it derives $(6.91 \times 10^{-1}, 1.71 \times 10^2, 1.99 \times 10^4, 3.37 \times 10^6)$ which implies that the part of length 9 will dominate the final estimated scores in this case of aggregation. Consequently, the performance tends to be similar to the one only taking length 9 into account. However, with the improved one, the result of multiplication is $(6.91 \times 10^{-1}, 1.71 \times 10^{-2}, 1.99 \times 10^{-3}, 3.37 \times 10^{-4})$ which guarantees that the result of less length will contribute more to the final predictions. Naturally, it will yield more accurate recommendation using the improved weighting factor. Actually, each weighting factor must be less than the reciprocal value of the maximum entry of its corresponding power of the adjacency matrix. Because the proposed algorithm can achieve high performance only with length 3, the subsequent experiments for the proposed algorithm just take length 3 into account and we refer to this method as $Complex$.

Fig. 4 shows the comparative $hits\ rate$ and $coverage$ of the recommender algorithms introduced in Section III-A on MovieLens dataset. We denote $ItemBasedPear$ as the Item based collaborative filtering with Pearson correlation method for similarity measurement.

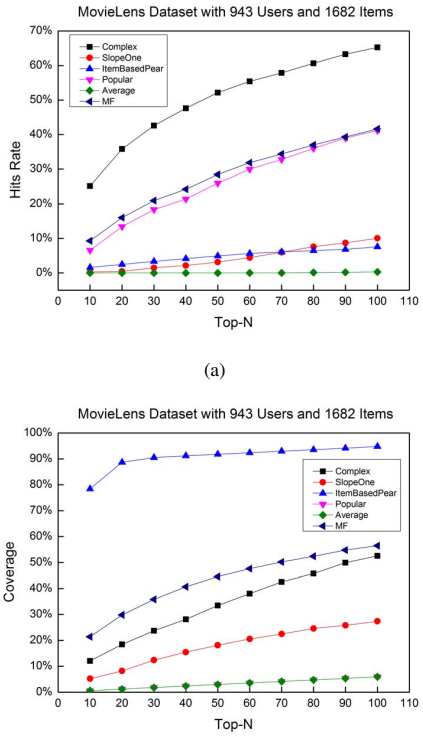


Fig. 4. The $hits\ rate$ and $coverage$ comparison of the comparison methods on MovieLens. (The Average and Popular are two non-personalized methods which recommend users the same items, so their coverages are the same with two overlapping curves.)

$Complex$ outperforms other methods by $hits\ rate$ and has relatively high $coverage$. $ItemBasedPear$ suffers from low $hits\ rate$, while its $coverage$ is very high. The reason is that it cannot make accurate recommendation when the dataset is sparse, which results in low $hits\ rate$. In this case, its $coverage$ becomes less meaningful, since a high coverage value is desirable only when the accuracy result is comparable. Moreover, it is surprising that the one which only recommends popular items to users obtained higher $hits\ rate$ than $ItemBasedPear$ and Slop One. It is because of that the popular items will have high probabilities to appear in the test set using the random partition method.

Fig. 5 illustrates the similar results on AppChina dataset. It can conclude that the proposed algorithm, $Complex$, not only has higher recommendation accuracy, but also presents better $coverage$.

Based on the basic link prediction approach based recommendation algorithm with complex number, we proposed a series of improved algorithms. The basic one just takes the number of paths and corresponding lengths between two nodes into account, which is based on such an assumption that the more the paths and the shorter the path lengths are, the closer these two nodes will be. However, it does

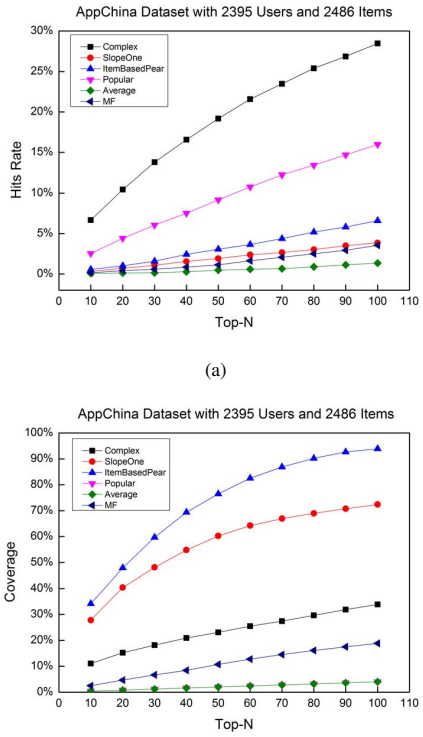


Fig. 5. The $hits\ rate$ and $coverage$ comparison of the comparison methods on AppChina. (The Average and Popular are two non-personalized methods which recommend users the same items, so their coverages are the same with two overlapping curves.)

not distinguish the different importance of paths with the same length. Intuitively, the paths with low degree (or popularity) nodes should contribute more to measure the closeness between two endpoints than those with high degree nodes. Here, we proposed $Complex\_Item$, $Complex\_User$ and $Complex\_User\&Item$ algorithms, respectively. These improved algorithms are slightly different from the basic Complex algorithm in the adjacency matrix modelling, while the calculation of power of the adjacency matrix and the final recommendation are the same. We define $|N_u(i)|$ and $|N_i(u)|$ as the degree of user u and item i, respectively. Then, each entry (u, i) of the adjacency matrix can be formulated in Table IV.

TABLE IV. ENTRY COMPARISON OF COMPLEX NUMBER BASED ALGORITHMS.

Entry/ Algorithm	$Complex$	$Complex\_Item$	$Complex\_User$	$Complex\_User\&Item$
$like$	j	$\frac{j}{\sqrt{N_i(u)}}$	$\frac{j}{\sqrt{N_u(i)}}$	$\frac{j}{\sqrt{N_i(u)\cdot N_u(i)}}$
$dislike$	$-j$	$-\frac{j}{\sqrt{N_i(u)}}$	$-\frac{j}{\sqrt{N_u(i)}}$	$-\frac{j}{\sqrt{N_i(u)\cdot N_u(i)}}$

Fig. 6 shows the experimental results of four complex number based algorithms by $hits\ rate$ and $coverage$ as the number of recommended items for users ranges from 10 to 100 on MovieLens dataset (the similar results on AppChina dataset are not given). These algorithms have the same property that the $hits\ rate$ and $coverage$ increase as the growth of the number of recommended items and the improved ones achieve higher $hits\ rate$ than $Complex$ method. We can also find that the recommended items will be more relevant to users when user and item degree are taken into account, which is derived from that $Complex\_User\&Item$ method outperforms other methods by $hits\ rate$.

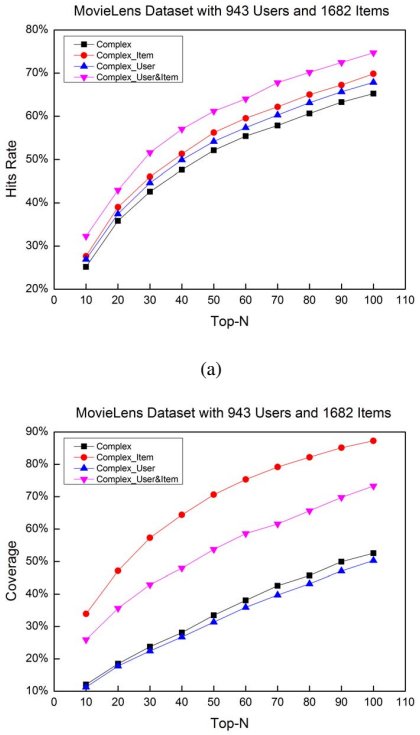


Fig. 6. The $hits\ rate$ and $coverage$ comparison of the complex number based algorithms on MovieLens dataset.

Moreover, it can be figured out that $Complex\_Item$ obtains significantly better performance than $Complex\_User$ by both $hits\ rate$ and $coverage$, because user degree is less meaningful than item degree due to user subjectivity. In real-world cases, some users may have seen a lot of movies, but they do not like to give comments. Hence accounting their degrees small will be a big mistake. On the contrary, popular items will get more ratings, while the unpopular ones will get less. Therefore, item degree seems more believable. We should note that despite $Complex\_User\&Item$ method gets higher $hits\ rate$ than $Complex\_Item$, whose $coverage$ seems much poorer.

IV. Conclusions and Future Work

In this paper, we proposed a recommendation algorithm based on link prediction approach with the weights in the graph represented by complex number, which can efficiently distinguish the $similar$ between two users/items and the $like$ between user and item. Consequently, the previous link

prediction methods can reuse without any modifications. Extensively experimental results show that the proposed algorithm outperforms many popular algorithms on MovieLens and AppChina datasets by two quality metrics: $hits\ rate$ and $coverage$. Moreover, it is experimentally verified that the recommendation generated by less length of path can be more efficient. We also point out that aggregating the results from different lengths of path with the arbitrary weighting factor series may be poor. In order to improve the performance of the proposed algorithm, the user degree and item degree are taken into account to distinguish the different importance of paths with the same length. The experimental results are extremely good and support that item degree is more valuable than user degree.

As the power of matrix calculation is time and space consumption when the users and items grow, it is essential to parallelize this method so that it can run on AppChina.com website for application recommendation in the future. Moreover, the improved complex number based algorithms achieve relatively high performance, so it is no doubt that many other magical factors remain to be developed to make our proposed method more powerful.

ACKNOWLEDGMENT

This work was supported in part by Ministry of Science and Technology of China under National 973 Basic Research Program (No. 2013CB228206 and No.2012CB315801), National Natural Science Foundation of China (grant No. 61233016), and China NSFC A3 Program (No.61140320).

REFERENCES

[1] U. Hanani, B. Shapira, and P. Shoval, "Information filtering: Overview of issues, research and systems," *User Modeling and User-Adapted Interaction*, vol. 11, no. 3, pp. 203–259, 2001.

[2] N. J. Belkin, "Helping people find what they don't know," *Communications of the ACM*, vol. 43, no. 8, pp. 58–61, 2000.

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

[4] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[5] Y. Jiang, J. Shang, and Y. Liu, "Maximizing customer satisfaction through an online recommendation system: A novel associative classification model," *Decision Support Systems*, vol. 48, no. 3, pp. 470–479, 2010.

[6] K.-W. Cheung, J. T. Kwok, M. H. Law, and K.-C. Tsui, "Mining customer product ratings for personalized marketing," *Decision Support Systems*, vol. 35, no. 2, pp. 231–243, 2003.

[7] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," *Artificial Intelligence Review*, vol. 13, no. 5-6, pp. 393–408, 1999.

[8] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[9] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web*, 2007, pp. 291–324.

[10] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and 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.

[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*, 2001, pp. 285–295.

[12] J. Bobadilla, F. Ortega, A. Hernando, and J. Alcalá, "Improving collaborative filtering recommender system results and performance using genetic algorithms," *Knowledge-based systems*, vol. 24, no. 8, pp. 1310–1316, 2011.

[13] J. Bobadilla, F. Serradilla, and J. Bernal, "A new collaborative filtering metric that improves the behavior of recommender systems," *Knowledge-Based Systems*, vol. 23, no. 6, pp. 520–528, 2010.

[14] K. Choi and Y. Suh, "A new similarity function for selecting neighbors for each target item in collaborative filtering," *Knowledge-Based Systems*, vol. 37, pp. 146–153, 2013.

[15] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, pp. 109–132, 2013.

[16] H. Ma, T. C. Zhou, M. R. Lyu, and I. King, "Improving recommender systems by incorporating social contextual information," *ACM Transactions on Information Systems (TOIS)*, vol. 29, no. 2, p. 9, 2011.

[17] F. Xie, M. Xu, and Z. Chen, "Rbra: A simple and efficient rating-based recommender algorithm to cope with sparsity in recommender systems," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, 2012, pp. 306–311.

[18] F. Xie, Z. Chen, H. Xu, X. Feng, and Q. Hou, "Tst: Threshold based similarity transitivity method in collaborative filtering with cloud computing," *Tsinghua Science and Technology*, vol. 18, no. 3, pp. 318–327, 2013.

[19] F. Xie, Z. Chen, J. Shang, and G. C. Fox, "Grey forecast model for accurate recommendation in presence of data sparsity and correlation," *Knowledge-Based Systems*, 2014.

[20] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J. R. Wakeling, and Y.-C. Zhang, "Solving the apparent diversity-accuracy dilemma of recommender systems," *Proceedings of the National Academy of Sciences*, vol. 107, no. 10, pp. 4511–4515, 2010.

[21] Z. Huang, D. Zeng, and H. Chen, "A comparative study of recommendation algorithms in e-commerce applications," *IEEE Intelligent Systems*, vol. 22, no. 5, pp. 68–78, 2007.

[22] T. Zhou, J. Ren, M. Medo, and Y.-C. Zhang, "Bipartite network projection and personal recommendation," *Physical Review E*, vol. 76, no. 4, p. 046115, 2007.

[23] X. Li and H. Chen, "Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach," *Decision Support Systems*, vol. 54, no. 2, pp. 880–890, 2013.

[24] L. Getoor and C. P. Diehl, "Link mining: a survey," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 2, pp. 3–12, 2005.

[25] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[26] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering." in *SDM*, vol. 5, 2005, pp. 1–5.

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

[28] B. Sarwar, G. Karypis, J. Konstan, and 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, and 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 and 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, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 5–53, 2004.

[32] F. Cacheda, V. Carneiro, D. Fernández, and V. Formoso, "Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems," *ACM Transactions on the Web (TWEB)*, vol. 5, no. 1, p. 2, 2011.