Decision Support Systems 54 (2013) 880–890

Contents lists available at SciVerse ScienceDirect

Decision Support Systems

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

Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach

Xin Li [a,*], Hsinchun Chen [b]

[a] Department of Information Systems, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon Tong, Hong Kong
[b] Department of MIS, University of Arizona, 1130 East Helen St., Rm. 430, Tucson, AZ, USA

ARTICLE INFO

Article history:
Received 23 August 2011
Received in revised form 21 June 2012
Accepted 18 September 2012
Available online 4 October 2012

Keywords:
Recommender systems
Kernel-based methods
Link prediction
Bipartite graph
Collaborative filtering

ABSTRACT

Recommender systems have been widely adopted in online applications to suggest products, services, and contents to potential users. Collaborative filtering (CF) is a successful recommendation paradigm that employs transaction information to enrich user and item features for recommendation. By mapping transactions to a bipartite user–item interaction graph, a recommendation problem is converted into a link prediction problem, where the graph structure captures subtle information on relations between users and items. To take advantage of the structure of this graph, we propose a kernel-based recommendation approach and design a novel graph kernel that inspects customers and items (indirectly) related to the focal user–item pair as its context to predict whether there may be a link. In the graph kernel, we generate random walk paths starting from a focal user–item pair and define similarities between user–item pairs based on the random walk paths. We prove the validity of the kernel and apply it in a one-class classification framework for recommendation. We evaluate the proposed approach with three real-world datasets. Our proposed method outperforms state-of-the-art benchmark algorithms, particularly when recommending a large number of items. The experiments show the necessity of capturing user–item graph structure in recommendation.

1. Introduction

Consumers nowadays have an increasing amount of experience with online recommender systems, such as when buying books from Amazon, borrowing movies from Netflix, or setting up friend circles on Facebook. Compared with search engines, recommender systems apply information filtering mechanisms that may lead users to items they are not aware of or cannot access using a keyword search. Well-designed recommender systems save users' time, improve customer satisfaction [29], and promote sales [12]. To better capture users' interests and make effective recommendations, it is necessary to combine multiple models [35] and make effective use of different types of data, such as user information, item information, and transaction information (business transactions, browsing activities, review activities, etc.) [32,49]. The collaborative filtering (CF) recommendation paradigm models users' collaborative behaviors reflected in transactions and cross-recommends products to users.There are generally two types of recommendation tasks, predicting purchase vs. predicting rating. Transaction/purchase is essentially an implicit and coarse rating on preferring an item [22]. Furthermore, it does not differentiate the statuses of "unknown" vs. "unlike." Thus, the two tasks are quite different from each other according to their computational nature. In real-world e-commerce applications, there generally exists more transaction information than explicit rating information. In this research, we focus on the modeling and prediction of transactions.

Given a set of users U and a set of objects/items O, we represent transactions T between them as a user-object interaction graph $G=(V, E)$, where $V=U\cup O$ and $E=\{(u,o): u\in U, o\in O, u\rightarrow o\in T\}$. This is a bipartite graph since user nodes can connect only to item nodes and vice versa. Under this representation, recommendation is equivalent to link prediction between users and items based on the existing graph (of previous transactions). The graph representation reveals subtle relations between indirectly connected users and items. Several collaborative filtering heuristic algorithms have explored the structure of user–item interaction graphs to improve recommendation performance [27,70]. However, few learning-based methods explicitly utilize the graph in constructing effective personalized recommendation models. Existing learning-based recommendation algorithms usually rely on explicit feature extraction, which is difficult to apply onto graph-structured data due to the requirements of superior computational capacity and extensive domain knowledge (to design features).

In this research, we propose a generic kernel-based machine learning approach of link prediction in bipartite graphs and apply it in recommender systems. We inspect nodes and links that are close to a focal user–item pair as its context to predict the possibility for the pair to interact, i.e., the possibility that the user may use/buy the item. We propose a novel graph kernel to capture the structure and user/item features in the context of focal user–item pairs and feed it into the one-class SVM algorithm to build the prediction

* Corresponding author. Tel.: +852 34424446; fax: +852 34420370.
E-mail addresses: xin.li@cityu.edu.hk (X. Li), hchen@eller.arizona.edu (H. Chen).

0167-9236/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.dss.2012.09.019

model. We examine the validity and computational efficiency of the graph kernel. We demonstrate the performance of this recommendation approach using three real-world datasets.

The paper is organized as follows. The second section reviews related studies on recommendation algorithms. The third section introduces the proposed graph kernel-based recommendation framework. The fourth section describes experiments on three real-world datasets and discusses the experimental results. The last section summarizes the findings and proposes directions for future research.

2. Literature review

2.1. Recommendation algorithms

There have been several survey papers on recommender system studies [3,24]. In this research, we focus on the utilization of graph structure in the design of recommendation algorithms and review previous studies according to the feature types and computational techniques.

First, previous studies generally use two types of features in designing the recommendation algorithms: collective local features and graph-related features.

Collective local features capture the collective characteristics of individual user/item information, such as a user's demographic characteristics, content of interest [32], item's specifications, transaction contexts (environment, time, etc.), and temporal usage patterns (which reflect user's characteristics). Collective local features directly show differences between people and the products they bought. Thus, researchers design similarity measures to cross-recommend similar products among similar users [5]. Researchers also proposed several probabilistic latent variable models on users' usage/purchase histories that group users/items to (hidden) classes to estimate the probabilities of user–item interactions [21].

Graph-related features highlight interactions between related users/items. By representing transactions using a user–item interaction graph, the subtle relations between indirectly connected users and items can be modeled by the graph structure. In light of the graph theory, topological characteristics of nodes on the graph were considered an indication of items' attractiveness and importance in the network, which can be used in recommendation [18]. Graph structure can be used to design similarity measures for cross-recommendation. The bipartite user–item graph can also be projected onto a unipartite user/item graph [70] to simplify the graph structure. Furthermore, some researchers construct graphs based on user/item similarities. Such artificial graphs can help alleviate the data sparsity problem [8]. In Webpage recommendation applications, a Webpage browsing graph has been employed to capture user behaviors and interest in certain contents [62].

Second, previous recommendation algorithms generally include two types of computational techniques: heuristic (i.e., memory-based) algorithms and learning-based (model-based) algorithms.

Heuristic algorithms are designed based on rules/measures provided by domain experts. A widely adopted heuristic is to recommend the most popular items (according to number of sales) to users. User/item similarity-based cross-recommendation is also a type of heuristic, which relies on designing appropriate similarity measures. In previous research, several similarity measures have been proposed based on user/item features [32]and transactional characteristics [5]. The most popular measures were the Pearson correlation coefficient [52] and cosine-based similarity [55].

User/item similarities have been combined with the interaction graph structure to capture information in connected nodes for link prediction [36]. Liben-Nowell and Kleinberg [43] explored the algorithms that iteratively compare nodes and update node similarities to a global node similarity measure. Taking a graph view of transactions, several previous studies proposed using eigenvector-based node ranking algorithms to rank items for recommendation. These algorithms are in general similar to PageRank [18,19,23] and HITS [27], which model influence of users and attractiveness of products based on the connectivity of the graph. Furthermore, Fouss et al. proposed that closer users and items on the interaction graph (as measured by conducting random walks between them) may have a higher probability of interacting [14,15]. Huang et al. [26] also conjectured that possible links should lead to some topological changes on the graph, such as on clustering coefficients.

Compared to heuristic algorithms, learning-based methods build models based on existing data instances. Several probabilistic models, such as probabilistic latent semantic analysis (PLSA) [21], have been proposed to learn from business transactions [50,67,68] to predict future purchases. It is a common practice for such models to design and capture hidden user classes according to purchase histories and use the classes to aid recommendation. Temporal information of user purchase histories can be captured with a maximum entropy model to better tackle the recommendation problem [28,48]. Other research has explored finding the most effective features in business transactions that can be fed into mature machine learning models to do recommendation. For example, the probabilistic relational model (PRM) [17,45] was utilized to make predictions based on product features [10]. The regression model has been used with product features [60] and user rating features [2] to make recommendations. The SVM algorithm was applied on product features [64] and transaction context features, such as time, weather, etc., [4,46] to examine the probability that a product will be selected by users.

Due to its success in a recent Netflix contest, the matrix factorization method has attracted significant interest. Matrix factorization assumes that each item and user can be characterized by a set of factors whose inner product is the user's rating of the item [35], which are essentially hidden groups of users and items. These factors can be learned by minimizing the estimation error using the stochastic gradient descent or alternating least squares methods. Variations of matrix factorization techniques have been explored for computational efficiency and data characteristic concerns [16,34,47,53]. Notably, Hu et al. [22] have developed a method for matrix factorization on binary matrices, which better suits the purchase recommendation problem.

It is also possible to use graph-related features in the learning-based paradigm. Huang et al. [25] extended the PRM framework and defined rules to extract features on connected nodes in the user–item interaction graph to construct recommendation models. Yajima [65] used a Laplacian kernel to capture the positional relations among nodes on the graph and built one-class SVM models for each user to recommend items that are positionally closer to their previously bought items. Under an unsupervised learning paradigm, Reddy et al. [51] proposed to use a graph-based clustering algorithm to group similar users on the user–item graphs. The user groups help recommendation. Table 1 summarizes major research on recommendation algorithms. In general, a significant amount of effort focused on learning-based models using collective local features. Several studies explored the use of graph features in heuristics. There are limited studies using learning-based models on explicit graph data representations.

2.2. Link prediction in graphs

Using graph representations, a recommendation problem can be converted to a link prediction problem, which is an active research area in computer science. To our best knowledge, most link prediction research focused on unipartite networks, such as social networks, Webpages, and email networks. However, the recommendation problem is on bipartite networks.

In link prediction research, several heuristics have been proposed. For example, sociologists identified a strong homophily phenomenon in friendships [44]. This rationale led to a heuristic that uses actor similarity to infer social links. Liben-Nowell and Kleinberg [43] adopted and proposed several graph-based similarity measures for social

Table 1
A summary of previous recommendation algorithm studies.

Study	Feature[a]	Technique[b]	Notes
Resnick et al. 1994 [52]	L	H	Similarity-based
Sarwar et al. 2001 [55]	L	H	
Ahn 2008 [5]	L	H	
Kim et al. 2011 [32]	L	H	
Huang et al. 2004 [23]	G	H	Eigenvector-based node ranking
Griffith et al. 2006 [19]	G	H	
Gori and Pucci 2007 [18]	G	H	
Huang et al. 2007 [27]	G	H	
Zhou et al. 2007 [70]	G	H	
Fouss et al. 2007 [15]	G	H	Node position-based
Fouss et al. 2012 [14]	G	H	
Huang et al. 2007 [26]	G	H	Clustering coefficient-based
Hofmann 2004 [21]	L	L	Probabilistic model
Yu et al. 2004 [67]	L	L	
Zeng et al. 2004 [68]	L	L	
Polcicova and Tino 2004 [50]	L	L	
Iwata et al. 2008 [28]	L	L	
Getoor and Sahami 1999 [17]	L	L	Machine learning model + local features
Newton and Greiner 2004 [45]	L	L	
Vucetic and Obradovic 2005 [60]	L	L	
Adomavicius and Tuzhilin 2005 [3]	L	L	
Xu and Araki 2006 [64]	L	L	
Adomavicius and Tuzhilin 2011 [4]	L	L	
Funk 2006 [16]	L	L	Matrix factorization
Paterek 2007 [47]	L	L	
Salakhutdinov and Mnih 2008 [53]	L	L	
Koren 2008 [34]	L	L	
Hu et al. 2008 [22]	L	L	
Koren et al. 2009 [35]	L	L	
Huang et al. 2004 [25]	G	L	PRM + aggregative features
Yajima 2006 [65]	G	L	One-class SVM + Laplacian kernel
Reddy et al. 2002 [51]	G	L	Graph-based clustering

[a] L — collective local features; G — graph-related features.
[b] H — heuristic methods; L — learning-based methods.

network link prediction. Acar et al. [1] applied matrix factorization on one such measure, the Katz measure, to improve link prediction performance. In graph theory, the preferential attachment model for network evolution [6] can be used for link prediction, which predicts that high degree nodes will have higher probability to initiate new links.

Model-based methods have also been used in link prediction. The Markov chain model has been used to explore user navigation history for Webpage link prediction [54,71]. Cohn et al. [13] applied the PLSA algorithm on Webpage contents and hyperlinks for hyperlink prediction. As an extension of the PRM model, Taskar et al. [59] proposed the use of the Relational Markov Network (RMN) on link prediction. Hasan et al. [20] proposed a supervised learning framework that considers node proximity features, aggregated linkage features, and topological features for link prediction, which is also applied in [7]. Such a method can also be utilized in the recommendation problem. Wang et al. [61] adopted a logistic regression model and used topological and content-based similarity measures to predict co-authorship relations. Yu and Chu [66] adopted the Gaussian process approach and decompose link similarities to node-wise linear kernels to build a link prediction model. Kashima et al. [31] and Scripps et al. [57] took an optimization approach to assign same-link labels to links with similar node characteristics.

2.3. Kernel-based machine learning methods and graph kernels

In general, it is difficult to define features on the complicated structure of graphs. As an alternative machine learning framework, kernel-based machine learning provides a systematic way to deal with this problem. Compared to traditional feature-based methods, kernel-based methods do not require explicit feature generation. A kernel-based method relies on a kernel function that defines a similarity measure between data instances, $k: \chi \times \chi \rightarrow R$. The kernel function maps data from the input space χ to a feature space H (named reproducing kernel Hilbert space, RKHS), $\Phi(x): \chi \rightarrow H$, where the mapping function $\Phi(x)$ is not explicitly defined and meets the condition $k(x,x') = <\Phi(x), \Phi(x')>$. A kernel-based method also need a kernel machine, which is an algorithm that needs only kernel function values to learn patterns of the data instances in the feature space H. There are only limited kernel machines, such as Support Vector Machines (SVM), one-class SVM, kernel Fisher discriminant (KFD), and Support Vector Data Description (SVDD). However, given the different characteristics of applications, the performance of kernel-based methods is highly dependent on the selection and design of kernel functions [58].

In graph-structured data, graph kernels, including the Laplacian kernel [65], diffusion kernels [33], commute time kernel [14], and marginalized kernel [30], are designed to capture features of graphs. They have been widely used in bioinformatics studies on gene interaction networks or protein interaction networks. Previously, graph kernels have been applied on graph classification (to classify protein functions according to their molecular structures) [9,39]. In these models, graphs are generally decomposed into their sub-structures, such as nodes, links, and random walk paths. Graph similarities are thus defined by aggregating the sub-structures' similarities. By representing nodes on the graph with sub-graphs close to them, graph kernels have also been generalized to node classification (to classify patents according to citation networks) [40].

There are studies that simply use kernels as node similarity values in a heuristic paradigm to cross-recommend products to users. For example, Kunegis built graph kernels as linear, polynomial, and exponential combinations of a graph adjacency matrix and Laplacian matrix [37,38] and then employed the kernel values to indicate the possibilities of a link between node pairs. However, studies on graph kernels in a machine learning scheme for link prediction are still limited.

2.4. Highlights

The three streams of literature together shed light on the potential directions to improve product recommendation models. First, learning-based methods are one active approach in this area. As compared with heuristics, this approach generally has more stable performances across different datasets (and requires more computational power). As a link prediction problem, recommendation deals with (the matching of) node pairs on a network. Thus, a straightforward approach is to compile local features from pairs of nodes and employ mature machine learning algorithms to classify whether they may interact. In co-authorship link prediction, authors' publication status, their common keywords, and distances between authors in a network have been used as user features [20]. In a recommendation task, product features, such as textual features on product descriptions, have been employed [64].

While the use of local features currently dominates learning-based recommendation algorithms, the explorations of graph features in

heuristics show the potential of using graph features in a learning-based framework. Kernel-based methods and graph kernels enabled such a direction. Yajima [65] provides a basic example of this approach, which uses a Laplacian kernel to capture node similarities that are related to distances between nodes. Under a one-class SVM algorithm, positionally closer users and items are more likely to be predicted to interact. In recommendation, several graph kernels have also been used under a rule-based framework (where the model is not trained from data) [15]. Many of these kernels take a similar idea of node closeness. They usually estimate distances between nodes through commute time or transition probabilities of random walks. One obvious limitation of such a kernel design is that it can only work with nodes that are (indirectly) connected. For more effective use of the kernel-based machine learning paradigm, it is necessary to carefully design and select appropriate kernels for recommendation.

Previous studies have provided many successful examples of the design of graph kernels. One effective approach is to inspect structural commonality of different parts of a network with the help of previous graph comparison kernels. For example, in patent classification [40], a node (patent) was accompanied by its two-level neighbors as a sub-graph and classified together into different categories. To compare such neighborhoods, the study decomposed sub-graphs into random walk paths and conducted pairwise comparisons of random walks paths. In this design, random walks are not used as a distance measure but simply to represent sub-structures of graphs. Such a design highlights the interdependency between nodes in a graph. However, to use similar design principles in recommendation, significant effort must be put into re-designing the kernel.

In short, to fill the research gap of limited learning-based methods on graph-related features in recommendation, we take a kernel-based approach [65] and develop graph kernels for modeling. Different from the designs based on node distance [15], we explore structural commonality between different parts of a network [40] in the kernel design. We aim to contribute to the design of an effective graph kernel that suits the requirement of the recommendation task.

3. Methodology

In recommendation, the task is to estimate the probability that user u and item o will interact. Here, we inspect the users and items close to u and o on the graph and consider these as the context to judge the relationship between u and o. We conjecture that user–item pairs with a topologically similar context may have a similar probability to have (or not have) an interaction. Based on their context characteristics, we can classify user–item pairs into two groups (positive and negative). The major task in this process is to design a graph kernel comparing contexts of user–item pairs.

3.1. A graph kernel-based recommendation framework

Fig. 1 shows the four steps in our graph kernel-based framework. 1) In the graph and feature extraction step, we construct the user–item graph from transaction histories. We also extract features describing nodes (users and items) from the data. 2) In the graph kernel construction step, we design a kernel $k()$ on user–item pairs based on their context structure and features. In kernel-based methods, the kernel design is the most essential module for prediction. 3) In the model building step, a classifier is built to separate potential links from impossible links. In the recommendation task we investigate, all known data instances are the transactions or interactions that have happened. We cannot differentiate whether unhappened transactions are negative data instances or transactions that will happen in the future. Thus, we take a one-class SVM algorithm [56] which looks for a hyper-plane to separate positive data instances from the origin of RKHS to deal with this task. As shown in Fig. 1, the points represent data instances and the line represents the hyper-plane. In this model, the data instances on the same side of origin as the hyper-plane are predicted to be negative (i.e., impossible to exist), and those on the opposite side of origin from the hyper-plane are predicted to be positive. In previous research, two-class classification was also applied [25], where unhappened transactions are sampled and considered as negative instances. However, the sampling process in this practical approach lacks sound theoretical justification. Thus, we choose the one-class classification method in this study. 4) In the prediction step, we rank items for each user according to predicted confidences of user–item interactions (so that we can provide top N recommendations). To make such a ranking, we use the Euclidean distance from data instances (i.e., user–item pairs) to the classification hyper-plane as the estimated confidence, where data instances farther from the hyper-plane (on the side opposite of the origin) have higher probability to occur. Given all support vectors $\{\overline{u_1o_1}, \overline{u_2o_2}, \ldots\}$ identified by the one-class SVM classifier, the confidence score of an instance $\overline{uo}$ is calculated as: $f(\overline{uo}) = \sum_i a_i k(\overline{uo}, \overline{u_io_i}) + b$, where a_i and b are estimated by the SVM algorithm. Such a measure is proportional to several classification confidence probability estimation measures for SVM [63]. Since we only care about ranks of items in recommendation, the measure is sufficient for this research.

3.2. Graph kernel design

Our designed kernel measures similarities of user–item pairs based on their related nodes/links, which utilizes the structure of the user–item graph. To capture the structural features, we generate random walk paths on the user–item graph as a simplified representation of the focal user–item pair's context. Since we work on a link prediction task, we use only the random walk paths starting from the focal user–

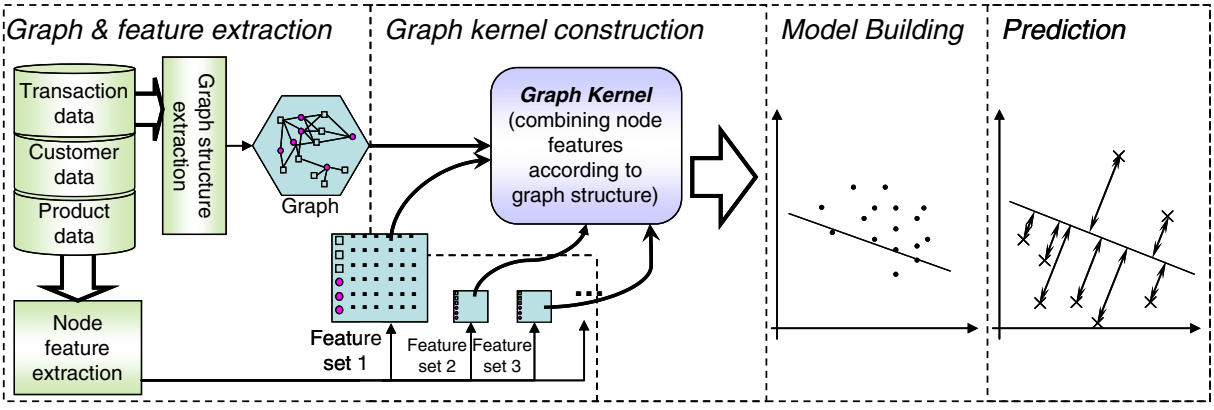

Fig. 1. A graph kernel-based recommendation framework.

item pair to other parts of the graph, which is a major difference between our design and previous research [9,40]. Fig. 2a shows an example context of a user–item pair $\overline{uo}$. In Fig. 2b, we present a transformed representation to show how to generate random walks from u and o. (Some nodes are presented more than once for explanation.) Examples of the random walk paths are shown in Fig. 2c.

Given a focal user–item pair $\overline{uo}$, a generated random walk path h on the pair can be represented as a sequence: $h = n_x^u \leftarrow \cdots n_2^u \leftarrow n_1^u \leftarrow u - o \rightarrow n_1^o \rightarrow n_2^o \cdots \rightarrow n_y^o$, where n_i^u ($i=1$ to x) and n_i^o ($i=1$ to y) are the mixture of user and item nodes on the u and o side of the path, respectively. (u and o can also be represented as n_0^u and n_0^o). The random walks connect the focal user–item pair to related users/items. In this research, we utilize only the random walk paths within a certain length to focus on the most relevant nodes. Features from such nodes are accumulated for the focal user–item pair's prediction. The generated random walks are associated with a probability of occurrence. Assuming the random walks jump from one node to its neighbors (or stop) with a transit probability p_t (or stop probability p_s), the path h has a probability of occurrence:

$$P(h|G) = p_s(n_x^u)p_t(n_x^u|n_{x-1}^u)\cdots p_t(n_2^u|n_1^u)p_t(n_1^u|u)p_t(n_1^o|o)p_t(n_2^o|n_1^o) \cdots p_t\left(n_y^o\middle|n_{y-1}^o\right)p_s\left(n_y^o\right).$$

Based on all generated random walk paths, we define the graph kernel, i.e., similarities of user–item pairs, as the sum of pairwise similarities between their related random walk paths weighted by the paths' probability of occurrence: $k_g(\overline{uo},\overline{u'o'}) = \sum_{\overline{uo}\subset h}\sum_{\overline{u'o'}\subset h'}\left[k_{path}(h,h')P(h|G)P(h'|G)\right]$, where $k_{path}(h,h')$ is the similarity between random walk paths. Generally speaking, the probability of occurrence reflects the strength of connections between the focal user–item pair and other users and items on the path. The nodes farther from focal user–item pairs will be involved with fewer and longer random walk paths and have smaller weight and impact on the focal user–item's link prediction.

To assess $k_{path}(h,h')$, we decompose the paths and compare their matching nodes and links according to their sequence. In this process, we always match focal user–item pairs between random walk paths. If two random walk paths do not have a one-to-one mapping after matching their focal user–item pairs (for example, without an equal length), we simply deem their similarity as 0. Otherwise, we define

$$k_{path}(h,h') = k_{node}(n_x^u,n_x^u)\times k_{node}(n_{x-1}^u,n_{x-1}^u)\times\ldots\times k_{node}(u,u') \times k_{node}(o,o')\times\ldots\times k_{node}\left(n_{y-1}^o,n_{y-1}^o\right)\times k_{node}\left(n_y^o,n_y^o\right),$$

where $k_{node}()$ is the similarity (kernel) between nodes. (Link features can also be used to assess random walk path similarities. However, most applications have limited features on interactions. Thus, we focus on node features in this research.)

Node similarity, $k_{node}()$, should be defined based on user/item features according to application and domain knowledge. It should be noted that the appropriate node features may vary across applications and interfere with graph structure. We leave the systematic exploration of effective node features to future research. In general, the selection of such features should highlight the factors that reflect/affect users' decisions, such as user age, education level, product category, etc. The representation of such features in a kernel form depends on characteristics of data; for example, a linear kernel can be applied on categorical data or textual data, and a Radial Basis Function (RBF) kernel can be applied on numerical data.

Finally, we normalize the graph kernel, which in general leads to a better prediction performance: $k(\overline{uo},\overline{u'o'}) = k_g(\overline{uo},\overline{u'o'})/\sqrt{k_g(\overline{uo},\overline{uo})k_g(\overline{u'o'},\overline{u'o'})}$.

3.3. Graph kernel characteristics

Computing the graph kernel by enumerating all random walk paths is computationally expensive. We therefore introduce a matrix formulation of the graph kernel calculation. For ease of discussion, we assume the adjacency matrix of the user–item interaction graph is $A=\{A_{i,j}\}$. The transition probability matrix is $M=\{M_{i,j}\}=\{p_t(j|i)\}$ and the stopping probability matrix is $Q=\{Q_{i,j}\}=\{p_s(j)\}$.

Proposition 1. The graph kernel K can be decomposed to the product of two random walk kernels starting from the user and item of focal user–item pairs.

Proof: Before normalization

$$k_g(\overline{uo},\overline{u'o'}) = \sum_{uo\subset h}\sum_{u'o'\subset h'}\left[k_{path}(h,h')P(h|G)P(h'|G)\right]$$

$$= \sum_{\substack{\overline{uo}\subset h;\overline{u'o'}\subset h';\\ h \text{ matches } h'}}\left[\prod_{i=1}^{x}k_{node}(n_i^u,n_i^u)\times k_{node}(u,u')\times k_{node}(o,o') \times\prod_{i=1}^{y}k_{node}(n_i^o,n_i^o)\times p_s(n_x^u)\times\left(\prod_{i=1}^{x}p_t(n_i^u|n_{i-1}^u)\right)\times\left(\prod_{i=1}^{y}p_t(n_i^o|n_{i-1}^o)\right)\times p_s\left(n_y^o\right) \times p_s\left(n_{y'}^u\right)\times\left(\prod_{i=1}^{x}p_t(n_i^u|n_{i-1}^u)\right)\times\left(\prod_{i=1}^{y}p_t(n_i^o|n_{i-1}^o)\right)\times p_s\left(n_{y'}^{o'}\right)\right]$$

$$= \sum_{\substack{\overline{uo}\subset h;\overline{u'o'}\subset h';\\ h \text{ matches } h'}}\left[\prod_{i=1}^{x}k_{node}(n_i^u,n_i^u)\times k_{node}(u,u')\times p_s(n_x^u)\times\prod_{i=1}^{x}p_t(n_i^u|n_{i-1}^u)\times p_s(n_{x'}^u) \times\prod_{i=1}^{x}p_t(n_i^{u\prime}|n_{i-1}^{u\prime})\times k_{node}(o,o')\times\prod_{i=1}^{y}k_{node}(n_i^o,n_i^{o\prime})\times\prod_{i=1}^{y}p_t(n_i^o|n_{i-1}^o)\cdot \times p_s\left(n_y^o\right)\times\prod_{i=1}^{y}p_t(n_i^{o\prime}|n_{i-1}^{o\prime})\times p_s\left(n_y^{o\prime}\right)\right].$$

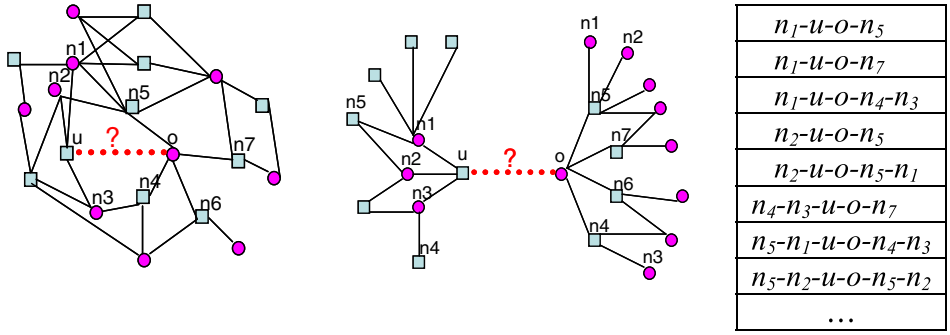

Fig. 2. Context of a user–item pair.

We define $h_u = u \to n_1^u \to n_2^u \ldots \to n_x^u$ and $h_o = o \to n_1^o \to n_2^o \ldots \to n_y^o$, which are the random walk paths starting from u and o, respectively.

$$k_g(\overline{uo}, \overline{u'o'}) = \sum_{h_u; h'_u; h_o; h'_o} \left[k_{path}(h_u, h'_u) P(h_u|G) P(h'_u|G) \times k_{path}(h_o, h'_o) P(h_o|G) P(h'_o|G) \right]$$

$$= \sum_{h_u; h'_u;} \left[k_{path}(h_u, h'_u) P(h_u|G) P(h'_u|G) \right] \times \sum_{h_o; h'_o} \left[k_{path}(h_o, h'_o) P(h_o|G) P(h'_o|G) \right].$$

We define $k_{half}(u, u') = \sum_{h_u; h'_u;} \left[k_{path}(h_u, h'_u) P(h_u|G) P(h'_u|G) \right]$.

We can then get $k_g(\overline{uo}, \overline{u'o'}) = k_{half}(u, u') k_{half}(o, o')$.

It can be easily shown that the normalized graph kernel $k()$ can be represented as the product of normalized $k_{half}()$, which can be represented as $k_{half\_norm}()$. □

k_{half} is identical to the context graph kernel in [41] that was used in a node classification problem. The kernel matrix $K_{half} = \{k_{half}()\}$ can be represented as:

$$K_{half} = \sum_{i=1}^{l} K_i, \quad K_1 = (M^*Q) K_{node} (M^*Q)^{\mathrm{T}}, \quad K_{i+1} = M(K_{node}{}^* K_i) M^{\mathrm{T}},$$

where $K_{node} = \{k_{node}()\}$, l is the maximum length of random walk paths considered, and operation * is the Hadamard product (i.e., entrywise product). In this formulation, each matrix K_i covers the random walk paths of length $= i$ and the features from nodes that are i step(s) away from the starting user (or item). It was proved that this matrix formula converges when n goes to ∞ given a uniform stopping probability p_s or a p_s larger than 0.5 [41].

Proposition 2. *The calculation of graph kernel K can be finished in $O(l|U|^2|O|)$ or $O(l|U||O|^2)$ time, whichever is in a larger scale, where $|U|$ is the number of users and $|O|$ is the number of items.*

Proof: Since the user–item interaction graph is a bipartite graph, the adjacency matrix A and the transition probability matrix M take the form of $\begin{pmatrix} 0 & X' \\ X & 0 \end{pmatrix}$, where X and X' indicate non-zero elements (without loss of generality, we assume user nodes are put in front of item nodes in these matrices). The kernel matrices, such as K_{node}, K_i, take the form of $\begin{pmatrix} Y & 0 \\ 0 & Y' \end{pmatrix}$, where Y and Y' indicate non-zero elements, since a user and an item do not have matched random walk paths or features and are not directly comparable. In the decomposed form of K_{half}, the matrix multiplication operation in calculating K_i can be simplified to the multiplication of non-zero sub-matrices. In a naive algorithm, it takes $O(2|U|^2|O| + 2|U||O|^2)$ time to finish. Furthermore, calculating K_i also needs $O(2|U||O|)$ or $O(|U|^2 + |O|^2)$ time for the Hadamard product operation. Thus, to calculate K_{half} takes at most $O(l|U|^2|O|)$ or $O(l|U||O|^2)$ time, whichever is in a larger scale. Calculating K from K_{half} takes another Hadamard product for $O([|U| + |O|]^2)$ time. Overall, the calculation of K can be finished in $O(l|U|^2|O|)$ or $O(l|U||O|^2)$ time. If appropriate numerical techniques are applied, the exponent may be further reduced. □

It should be noted that according to previous network analysis studies, l can be a small number to cover a large network. Thus, it can be considered as a constant scale in real applications.

Proposition 3. The graph kernel K is positive semi-definite and thus a valid kernel.

Proof: Since K_{node} is a selected valid kernel, for any vector $\alpha \in R^{|U|+|O|}$, $\alpha K_{node} \alpha^T \geq 0$. For any vector $\alpha \in R^{|U|+|O|}$, $\alpha K_1 \alpha^T = \alpha(M*Q) K_{node} [\alpha\ (M*Q)]^{\mathrm{T}}$. Considering $\alpha(M*Q)$ is a vector $\in R^{|U|+|O|}$, $\alpha K_1 \alpha^T \geq 0$. Thus, K_1 is a valid kernel. Since the Hadamard product is entrywise multiplication, according to the closure properties of kernel functions, $K_{node} * K_1$ is a valid kernel. By applying these two rules iteratively, it is easy to prove $K_{i+1} = M(K_{node} * K_i) M^{\mathrm{T}}$ is a series of valid kernels. The sum of these kernels, K_{half}, is also a valid kernel. K_g is the tensor product of two valid K_{half} kernels and is positive semi-definite. Last, normalization on K_g does not change the validity of the kernel function. Thus, K is positive semi-definite and a valid kernel. □

In our proposed framework, the Euclidean distance between a data instance (i.e., user–item pair) and the classification hyperplane built by one-class SVM is used as the confidence score to predict the probability of the user–item's interaction. Given support vectors $\{\overline{u_1 o_1}, \overline{u_2 o_2}, \ldots\}$ identified by the one-class SVM, this confidence score is: $f(\overline{uo}) = \sum_i a_i k(\overline{uo}, \overline{u_i o_i}) + b$, where a_i and b are parameters estimated by SVM. Since $k(\overline{uo}, \overline{u'o'}) = k_{half_n orm}(u, u') k_{half_n orm}(o, o')$, $f(\overline{uo}) = \sum_i a_i k_{half_n orm}(u, u_i) k_{half_n orm}(o, o_j) + b$.. In recommendation, we always compare different products for individual customers. Thus $a_i k_{half\_norm}(u,u_i)$ does not change for each user in each recommendation. The rank of items depends on their similarities with items in the support vectors $k_{half\_norm}(o,o_i)$. The difference between users is captured by $a_i k_{half\_norm}(u,u_i)$, in which the importance of each support vector is moderated by their similarity to the focal user $k_{half\_norm}(u,u_i)$. Since $k_{half\_norm}()$ depends on both node (local) features and graph structure, our proposed recommendation model finds representative reference users with similar contexts to the focal user and makes recommendations according to reference users' previous purchases.

4. Experimental study

4.1. Datasets

In order to show the algorithms' performance on different data characteristics, we use three datasets from quite different application settings to evaluate the proposed framework.

An online book retail dataset obtained from a major Chinese online bookstore in Taiwan [26]. The dataset includes 5 years of transactions of 2000 randomly selected users, involving 9695 books and 18,771 transactions.

An online clothing retail dataset provided by a leading U.S. online clothing merchant [27]. The dataset includes 16 million online transactions from a 3-month period, involving 4 million households and 128,000 items.

A book rating dataset collected from the Book-Crossing community [72]. The dataset reports 278,858 users' 1,149,780 ratings on 271,379 books. Since this research focuses on exploiting transaction information, we treat each rating as a transaction by assuming that a user buys a book before reading and rating it.

For meaningful testing, we include only users with 5 to 100 transactions. This constraint results in 851 users for the book retail dataset. For comparison purposes, we sample 1000 users with 5 to 100 items from the other two datasets. Table 2 reports the descriptive statistics of the reduced datasets. The two sampled book datasets show closer average purchases per user while the sampled clothing dataset and the book rating dataset have closer numbers of sales per item. Overall, the sampled book retail dataset is much denser than the other two. The reduced datasets were split into 80% training data and 20% testing data according to transaction time. Items that appeared only in the testing data, which cannot be predicted by any algorithm, are removed from the study.

4.2. Evaluation metrics

In the task of predicting transactions/purchases, the RMSE measure for rating prediction evaluation cannot be applied. To evaluate

Table 2
Dataset statistics.

Dataset	# of users	# of items	# of transactions	Avg. purchases/user	Avg. sales/item
Book retail	851 (~2000)	8566 (~9700)	13,902 (~18,000)	16.34 (~9)	1.62 (~1.86)
Clothing retail	1000 (~4 million)	7328 (~128,000)	9332 (~16 million)	9.33 (~4)	1.27 (~125)
Book rating	1000 (~280,000)	15,578 (~270,000)	19,329 (~1 15 million)	19.33 (~4.12)	1.24 (~4.24)

The numbers in parentheses are the statistics on the original dataset.

recommendation performance, we have each algorithm generate a ranked list of items for each user and compare the recommendations with actual transactions in the testing data using three types of measures.

- We use precision, recall, and F-measure to measure the algorithms' performance in top 10 recommendations [49].

$$\text{Precision} = \frac{\text{Number of recommended products that match with the future purchases}}{\text{Total number of recommendedproducts}}$$

$$\text{Recall} = \frac{\text{Number of recommended products that match with the future purchases}}{\text{Total number of products with future purchases}}$$

$$\text{F}-\text{measure} = 2*\text{Precision}*\text{Recall}/(\text{Precision} + \text{Recall})$$

- We use the rank score to measure top recommendations from the algorithms [49].

$$\text{Rank score } RS = 100\frac{\sum_i RS_i}{\sum_i RS_i^{\max}}, \text{ where } RS_i = \sum_j \frac{will\_buy(i,j)}{2^{(j-1)/(halflife-1)}} \text{ and}$$

$$will\_buy(i,j) = \begin{cases} 1, & \text{if the } j\text{th recommended product is in customer } i\text{'s future purchase list} \\ 0, & \text{otherwise.} \end{cases},$$

where the parameter *halflife* is the rank of the item that has a 50% likelihood of being read by users, which was set at 2 following previous research [27]. Under this parameter setting, the rank score is affected mainly by the first couple of items on the list.

- We inspect the ROC curve of the algorithm, which visualizes the recommendation performance for a large number of recommendations. Due to the unique requirement of recommendation, we have slightly revised the ROC curve. Specifically, we replace the X axis from false positive rate to the number of recommendations. Since these two variables are highly correlated, this change does not change the shape of the curve much. However, the figure is more straightforward for understanding the performance for different numbers of recommendations.

4.3. Experimental procedure

Our proposed algorithm is a graph kernel-based machine learning approach. In the evaluation, we compare our proposed approach with state-of-the-art benchmark algorithms and the kernel-based machine learning algorithms that do not effectively use graph information.

For the first set of experiments, we select seven state-of-the-art benchmark algorithms:

1) A user-based algorithm that cross-recommends items to similar users (user similarity is based on the items they bought before) [49]. The algorithm first computes a user similarity score based on their purchased items. A higher similarity score indicates users have purchased a larger set of common products. Then, the occurrences of items in previous purchases are aggregated while weighted by their similarity to the focal user. Items with a larger score are recommended to the focal user.
2) An item-based algorithm that recommends items that are similar to users' previously purchased items (item similarity is based on the users who bought the item) [49]. The algorithm first computes an item similarity score based on users who purchased them. A higher similarity score indicates items share a larger set of common users. Then, for each user, the items that are similar to all his/her previous purchases are aggregated while weighted by the similarity score. Items that are most similar to previous purchases of the focal user are recommended.
3) An item popularity algorithm that recommends items according to their sales/access volumes. Items with higher previous sales volume are recommended to all users.
4) A spreading activation algorithm [23], which is a PageRank-like node ranking algorithm. In this approach, consumers and products are represented as nodes in a graph, each with an activation level μ_j. To generate recommendations for user u, the node is first activated and then the effect is propagated to other parts of the graph with given probability. When the process converges, the final activation level (on items) indicates their probability of being selected by focal user u.
5) A link analysis algorithm [27], which is a HITS-like node ranking algorithm. This algorithm considers a link between a user u and an item o indicates that o represents part of u's interest and u partially represents o's user base. Thus, an item representativeness score (with respect to each user) and a user representativeness score (with respect to each item) can be calculated in the same fashion as the authority and hub scores in the HITS algorithm, where the sum of the item representativeness scores of the products linked to a user gives the user representativeness score and vice versa. The item representativeness score is used to generate recommendations for the focal user.
6) A matrix factorization algorithm considering user and item bias [35,47]. This algorithm decomposes the user–item adjacency matrix (a matrix of size $|U|\times|O|$) to the multiplication of two sets of vectors, which maps both users and items to a joint latent factor space. It is closely related to the singular value decomposition (SVD) in that it decomposes the user–item matrix. To address practical concerns, previous research has developed the stochastic gradient descent algorithm and also considers that bias may exist on individual users and items. This basic implementation is adopted in our experiments.
7) A binary matrix factorization algorithm [22]. This algorithm considers the fact that unhappened transactions may be due to "unknown" instead of "unlike." It gives different weights on positive and negative elements in binary matrices when conducting optimization to conduct SVD. We implement the paper's iterative optimization algorithm as a benchmark.

In our experiments, the parameters of all algorithms are optimized for their best performance. Since all these benchmark algorithms use only transaction information, for a fair comparison we only use transaction information for our proposed graph-based approach in this experiment.

For the second set of experiments, we employ the same kernel-based machine learning framework on different features. The purpose is to show the ability of our proposed approach in making use of graph structure and aggregating node information. The first benchmark is based solely on user's and item's local features: $k_{local}(\overline{uo},\overline{u'o'}) = k_{node}(u,u')k_{node}(o,o')$ [20]. This model directly judges probabilities of linkage based on user–item pair characteristics. The second benchmark considers nodes in the context equally as a set: $k_{set}(\overline{uo},\overline{u'o'}) = \sum_{\substack{i \text{ is within } k \\ \text{steps from } \overline{ui}}} \sum_{\substack{i' \text{is within } k \\ \text{steps from } \overline{u'i'}}} k_{node}(i,i')$. This kernel conducts pairwise comparison on two sets of nodes and sums node similarity as the overall similarity. In these two algorithms, node

Table 3
User and item features.

Dataset	Category	Feature	Type	Kernel
Book retail	User	Age	Numerical	RBF
		Education level	Numerical	RBF
	Item	Book title	Textual	Linear
		Keywords	Textual	Linear
		Introduction	Textual	Linear
Clothing retail	Item	Product category	Categorical	Linear
		Description	Textual	Linear
Book rating	User	Location	Categorical	Linear
		Age	Numerical	RBF
	Item	Book author	Categorical	Linear

features are critical for inferences. Thus, we use such features for all algorithms in this set of experiments. Table 3 shows the local features we selected in the three datasets and the kernels we choose to represent such features. As we have explained, the selection of node features and kernel representation is not a trivial task. In our experiments, we choose the features and kernels that lead to a better performance on local feature models, so that our model is compared with the best scenario of baseline methods. We also optimize the size of the neighborhood for the set kernel, which is two steps from the focal user–item pair. We separate the experiments on user information and item information. While it is possible to include both of them in a model, node feature selection is also (then) required. We leave the optimization of appropriate node features (considering the interference of graph structure) to future research.

When implementing the graph kernel, we conduct 10 iterations ($l=10$), which means 10 steps of random walks from the focal user–item pair in both directions. According to our observation, the kernel matrix has converged after such number of iterations. In fact, in our testbed, the prediction performance becomes stable after about 3 steps of random walks. It is not sensitive to steps of random walks after that. We specify a uniform stopping probability, $p_s(\,)=1-\lambda$ ($0<\lambda<1$), to generate random walks on the user–item graph. We assume equal probability of jumping from one node to any of its neighbors, i.e., $p_t(j|i)=\lambda/d(i)$, where $d(i)$ is the number of i's neighbors. We roughly optimize parameter λ (from 0.1 to 0.9) using the training dataset (using 80% training data to build the model and 20% training data to test performance), which leads us to set λ at 0.9.

For all three kernel-based algorithms, we use a popular SVM package, libSVM [11], to build prediction models. The parameters for SVM are selected using the grid search tool provided by libSVM based on experiments on the training data.

In both experiments, the unconnected user–item pairs in the training dataset are examined and ranked. The most possible N links (according to each algorithm) are recommended for each user. The ranked list is compared with the actual transactions in the testing dataset for evaluation.

4.4. Results and discussion

Table 4 reports the performance[1] of the top 10 recommendations as compared with the models that do not take a kernel-based framework. The algorithms with the best performance according to pairwise t-tests are highlighted. In general, for top 10 recommendations, the proposed graph kernel is always in the group of best algorithms according to precision, recall, and F-measure. The link analysis algorithm, spreading activation algorithm, and user-based algorithm are also among the best algorithms in different datasets. In general, the algorithms' relative performances are different across datasets, especially for heuristic algorithms, which are not tuned for the dataset. This is common in machine learning research. Since it is almost impossible for one algorithm to succeed in all applications, we generally prefer algorithms that are consistently good across datasets (while searching for the best ones for each particular application). Obviously, our proposed graph kernel is a good algorithm on data with different distributions in terms of precision, recall, and F-measure. According to the rank score measure, which values the first couple of recommendations more than later recommendations, our proposed algorithm is slightly lower (statistically significant) than the top ones in different datasets, although the performance difference is minor. The matrix factorization, item popularity, and link analysis algorithms in general achieve a good performance on rank score.

Table 4
Performance comparison between graph-based algorithms.

Dataset	Algorithms	Precision	Recall	F-measure	RS
Book retail	User-based	0.0242	0.1165	0.0377	8.2882
	Item-based	0.0076	0.0396	0.0121	2.4338
	Item popularity	0.0258	0.1317	0.0405	**12.4843**
	Link analysis	**0.0280**	**0.1408**	**0.0439**	**11.8745**
	Spreading activation	0.0224	0.1110	0.0349	9.3618
	Matrix factorization	0.0255	0.1286	0.0399	**12.4843**
	Binary Matrix factorization	0.0110	0.0593	0.0177	3.0812
	Graph kernel	**0.0286**	**0.1461**	**0.0449**	11.2838
Clothing retail	User-based	0.0114	0.0778	0.0193	**4.5900**
	Item-based	0.0078	0.0597	0.0136	3.0354
	Item popularity	0.0062	0.0326	0.0100	1.4173
	Link analysis	**0.0124**	**0.0818**	**0.0209**	**4.7752**
	Spreading activation	0.0076	0.0513	0.0129	3.1005
	Matrix factorization	0.0058	0.0309	0.0094	1.3935
	Binary matrix factorization	0.0088	0.0649	0.0153	3.9898
	Graph kernel	**0.0131**	**0.0855**	**0.0219**	**4.1307**
Book rating	User-based	**0.0082**	**0.0269**	**0.0119**	**1.8260**
	Item-based	0.0052	**0.0203**	0.0079	**1.3080**
	Item popularity	0.0058	**0.0224**	0.0089	**1.9283**
	Link analysis	0.0065	**0.0225**	0.0096	**1.6260**
	Spreading activation	**0.0071**	**0.0264**	**0.0106**	**1.7307**
	Matrix factorization	0.0054	**0.0203**	0.0082	**1.9307**
	Binary matrix factorization	0.0061	**0.0245**	0.0093	**1.4029**
	Graph kernel	**0.0077**	**0.0295**	**0.0118**	**1.6251**

Bold values are not significantly different from the largest ones at 90% confidence interval.

Fig. 3 reports the ROC curves for the top 1000 recommendations made by the different models. It confirms that the graph kernel was among one of the best algorithms for a smaller number of recommendations (about 50 recommendations for the two book datasets and about 20 recommendations for the clothing dataset). Moreover, when a large number of predictions are needed, the graph kernel significantly outperforms all other methods on the ROC curve. Its performance is about twice that of the item-based algorithm and 20% to 50% better than other algorithms. The matrix factorization algorithm also has a better performance than other benchmarks on a large number of predictions.

Table 4 and Fig. 3 together show the advantage of different algorithms. In general, our proposed graph kernel shows absolute advantage on a larger number of recommendations. Its performance is always among the best on top 10 recommendations. Its performance is slightly lower than the best ones on the first couple of recommendations. Matrix factorization and item popularity methods are good at recommending items at the very top of the list. With appropriate parameters, the binary matrix factorization algorithm can outperform standard matrix factorization on

[1] The prediction performance may look numerically low as compared with some other applications, such as binary classification. However, considering that the task is to pick 10 from thousands of products, the performance is at a reasonable scale (and consistent with previous research under a similar experimental setting or in practice). Furthermore, in an e-commerce Website, the recommendation list can be updated every time a person clicks/refreshes a Webpage. Given the large number of customers and clicks on each Website, algorithms with the reported performance can bring significant amounts of revenue.

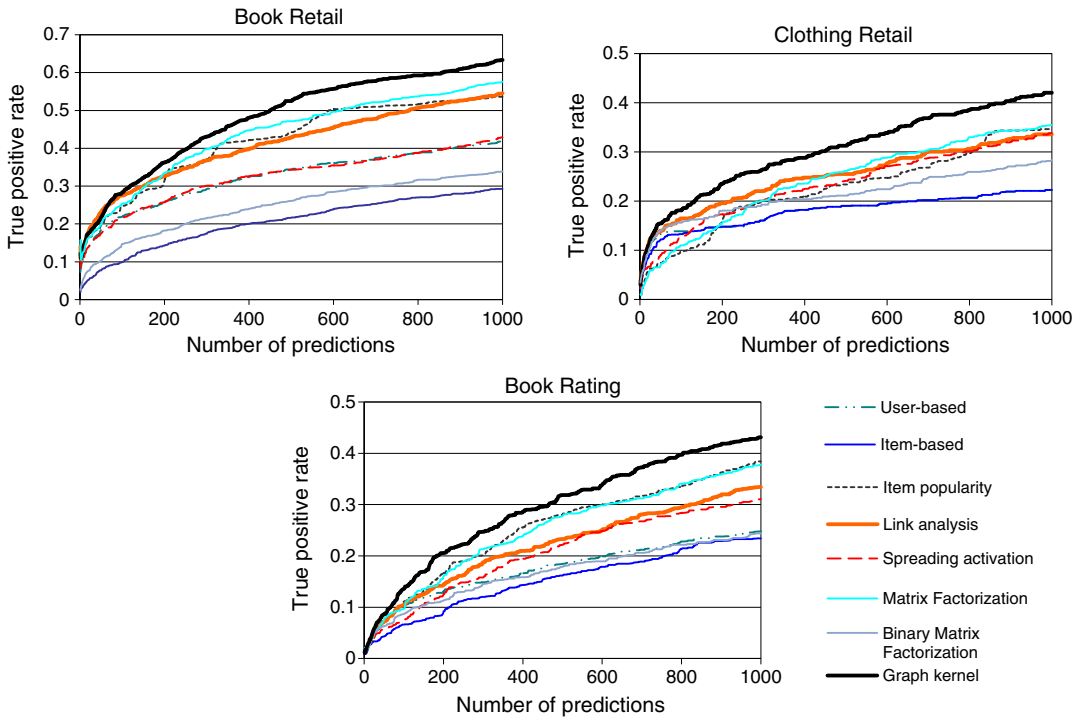

Fig. 3. ROC curves of top 1000 recommendations of graph-based algorithms.

top 10 recommendations. Other than the graph kernel, link analysis methods also perform well on predictions of the top 10 to 50 items.

In our proposed approach, the model is optimized to link prediction performance on the entire graph, i.e., differentiating overall positive vs. negative instances. It may provide more "high confidence" predictions to some users than others. When limited to a small number of recommendations, many of such predictions may be excluded, which limits its recommendation performance on rank score (as reflected in a couple of top predictions). On a larger number of predictions, its advantages can be fully exhibited. In real applications, a large number of predictions are always needed if e-commerce Websites want to update the recommendation list when users refresh the Webpages. Furthermore, repeat customers, who bring a large portion of profits, also require a larger number of recommendations. Our proposed graph kernel meets these requirements very well. The customers may be better served by the follow-up recommendations from our proposed approach. In practice, it is also beneficial to combine graph kernels with other algorithms to improve performance on a small number of recommendations.

Fig. 4 reports the ROC curves for the top 1000 recommendations made by the learning-based algorithms. (We omit the performance tables on top 10 recommendations due to space limitations. The three algorithms' performances on top 10 recommendations are generally distinguishable in the figure.) While the graph kernel performance may be improved with carefully selected node features, in the current setting the graph kernel significantly outperformed the set kernel and local features in most cases. The results clearly show the necessity of capturing the structure of the user–item graph in addressing the recommendation task. In the graph kernel, the features of individual nodes are accumulated to the focal user–item pairs following the links. The effective use of the user–item graph may cause its better performances. The set kernel ignores the graph structure, while the local feature-based kernel ignores all related nodes. These design deficiencies may have limited their prediction abilities.

When using item information on the book rating dataset, the local feature method initially achieved the best performance and then was outperformed by the graph kernel. In this dataset, the item information was author name. In the book rating application, the author of a book has a strong influence on users' ratings. A reader may choose a book purely because of the author. In comparison, different books rated by a reader may have much less topic/style similarities. Thus, co-rated books may have less prediction power than just local features on authors in these experiments.

5. Conclusions and future directions

In this paper, we present a graph kernel-based approach for recommendation. Considering the recommendation task as a link prediction problem in user–item interaction graphs, we define a graph kernel on the user–item pair's context and use its graph structure to infer whether a user may have a link with an item. We analyzed the graph kernel's characteristics including validity and computational efficiency. On three real-world datasets, our graph kernel-based approach shows improved recommendation performances over benchmark algorithms and kernel-based machine learning algorithms that make limited use of graph structure. Our proposed approach performed even better for recommending a large number of items.

We will explore how to extend this graph kernel-based approach to other link prediction and recommendation applications, such as information recommendation within mixed contents (i.e., related textual, image, and multimedia contents) and patent prior art search where patent examiners suggest additional references for patent applications. Such knowledge recommendation problems are critical to modern organizations [42,69]. We also plan to extend our work by combining multiple types of graphs into a model. For example, apart from the graph of user–item interactions, we will include social networks (between users) and/or citation networks (between items/contents) to address the link prediction and recommendation problem. We also plan to identify a systematic approach that will combine graph kernels with appropriate node information to fully exploit the ability of graph kernels in recommendation, which is not fully addressed in the scope of this paper.

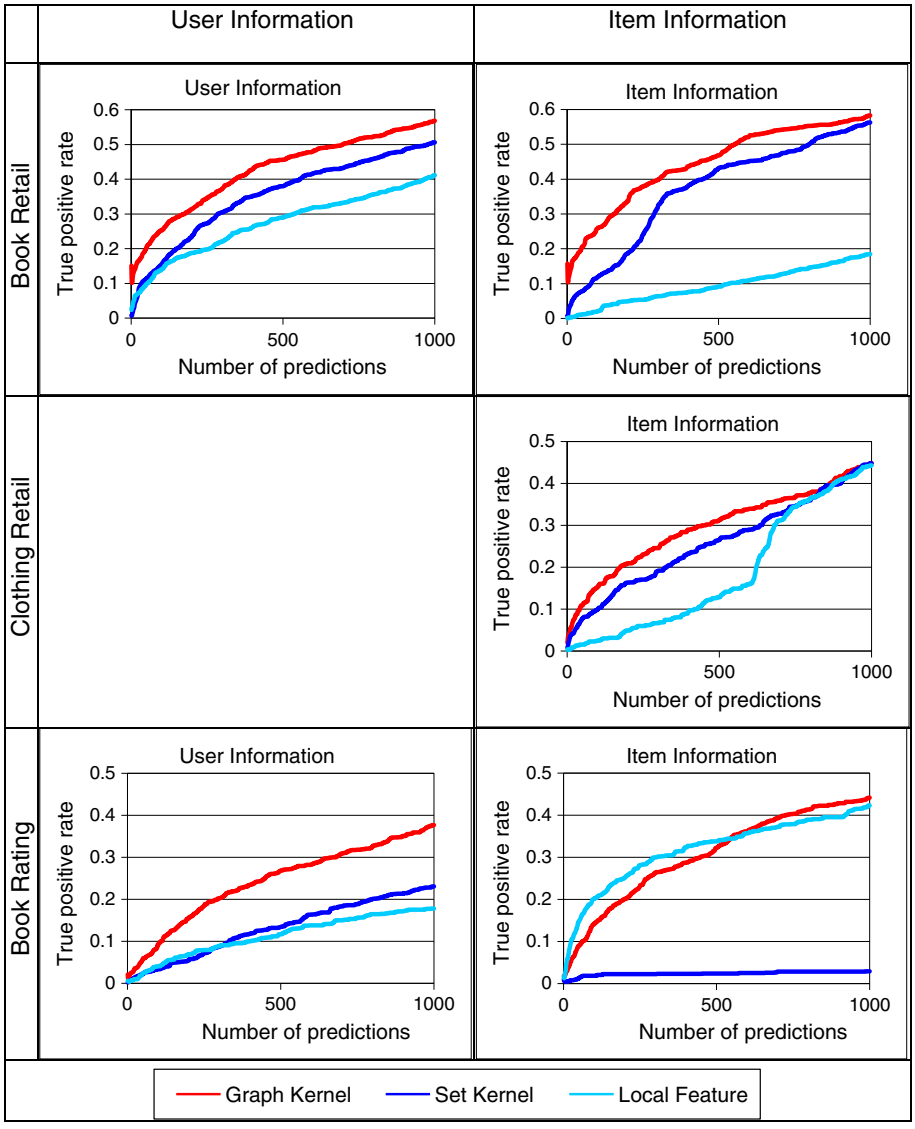

Fig. 4. ROC curves of top 1000 recommendations of learning-based algorithms.

Acknowledgments

This work was supported in part by: the City University of Hong Kong SRG-7002518, SRG-7002625, StUp-7200170, and the United States NSF IIS-0114011. All opinions in this article are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] E. Acar, D.M. Dunlavy, T.G. Kolda, Link prediction on evolving data using matrix and tensor factorizations, in: International Conference on Data Mining, 2009.
[2] G. Adomavicius, Y. Kwon, New recommendation techniques for multicriteria rating systems, IEEE Intelligent Systems (2007) 48–55.
[3] G. Adomavicius, 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 17 (2005) 734–749.
[4] G. Adomavicius, A. Tuzhilin, Context-aware recommender systems, in: F. Ricci, L. Rokach, B. Shapira, P.B.. Kantor (Eds.), Recommender Systems Handbook, Springer Science & Business Media, 2011.
[5] H.J. Ahn, A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem, Information Sciences 178 (2008) 37–51.
[6] A.L. Barabasi, R. Albert, Emergence of scaling in random networks, Science 286 (1999) 509–512.
[7] N. Benchettara, R. Kanawati, C. Rouveirol, Supervised machine learning applied to link prediction in social networks, in: International Conference on Advances in Social Networks Analysis and Mining, 2010, pp. 326–330.
[8] J. Bollen, M.L. Nelson, G. Geisler, R. Araujo, Usage derived recommendations for a video digital library, Journal of Network and Computer Applications 30 (2007) 1059–1083.
[9] K.M. Borgwardt, C.S. Ong, S. Schonauer, S.V.N. Vishwanathan, A.J. Smola, H.P. Kriegel, Protein function prediction via graph kernels, Bioinformatics 21 (2005) I47–I56.
[10] S. Boutemedjet, D. Ziou, A graphical model for context-aware visual content recommendation, IEEE Transactions on Multimedia 10 (2008) 52–62.
[11] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines, http://www.csie.ntu.edu.tw/~cjlin/libsvm 2001.
[12] K.W. Cheung, J.T. Kwok, M.H. Law, K.C. Tsui, Mining customer product rating for personalized marketing, Decision Support Systems 35 (2003) 231–243.
[13] D. Cohn, T. Hofmann, The missing link: a probabilistic model of document content and hypertext connectivity, Advances in Neural Information Processing Systems, 2001.
[14] F. Fouss, A. Pirotte, J.M. Renders, M. Saerens, Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation, IEEE Transactions on Knowledge and Data Engineering 19 (2007) 355–369.
[15] F. Fouss, K. Francoisse, L. Yen, A. Pirotte, M. Saerens, An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification, Neural Networks 31 (2012) 53–72.
[16] S. Funk, Netflix update: try this at home. Available from http://sifter.org/~simon/journal/20061211.html 2006.
[17] L. Getoor, M. Sahami, Using probabilistic relational models for collaborative filtering, in: International WebKDD Workshop, 1999.
[18] M. Gori, A. Pucci, ItemRank: a random-walk based scoring algorithm for recommender engines, in: International Joint Conference on Artificial Intelligence, 2007.
[19] J. Griffith, C. O'Riordan, H. Sorensen, A constrained spreading activation approach to collaborative filtering, in: The International Conference on Knowledge-Based Intelligent Information and Engineering Systems, 2006, pp. 766–773.
[20] M.A. Hasan, V. Chaoji, S. Salem, M. Zaki, Link prediction using supervised learning, in: Workshop on Link Analysis, Counter-terrorism and Security, 2006.

[21] T. Hofmann, Latent semantic models for collaborative filtering, ACM Transactions on Information Systems 22 (2004) 89–115.
[22] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: IEEE International Conference on Data Mining, 2008.
[23] Z. Huang, H. Chen, D. Zeng, Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering, ACM Transactions on Information Systems 22 (2004) 116–142.
[24] Z. Huang, W.Y. Chung, H.C. Chen, A graph model for e-commerce recommender systems, Journal of the American Society for Information Science and Technology 55 (2004) 259–274.
[25] Z. Huang, D. Zeng, H. Chen, A unified recommendation framework based on probabilistic relational models, in: 4th Annual Workshop on Information Technologies and Systems, 2004.
[26] Z. Huang, D. Zeng, H. Chen, Analyzing consumer-product graphs: empirical findings and applications in recommender systems, Management Science 53 (2007) 1146–1164.
[27] Z. Huang, D. Zeng, H. Chen, A comparative study of recommendation algorithms in e-commerce applications, IEEE Intelligent Systems 22 (2007) 68–78.
[28] T. Iwata, K. Saito, T. Yamada, Recommendation method for improving customer lifetime value, IEEE Transactions on Knowledge and Data Engineering 20 (2008) 1254–1263.
[29] Y.C. Jiang, J. Shang, Y.Z. Liu, Maximizing customer satisfaction through an online recommendation system: a novel associative classification model, Decision Support Systems 48 (2010) 470–479.
[30] H. Kashima, K. Tsuda, A. Inokuchi, Marginalized kernels between labeled graphs, in: The 20th International Conference on Machine Learning, 2003.
[31] H. Kashima, T. Kato, Y. Yamanishi, M. Sugiyama, K. Tsuda, Link propagation: a fast semi-supervised learning algorithm for link prediction, in: SIAM Data Mining Conference, 2009.
[32] H.-N. Kim, I. Ha, K.-S. Lee, G.-S. Jo, A. El-Saddik, Collaborative user modeling for enhanced content filtering in recommender systems, Decision Support Systems 51 (2011) 772–781.
[33] R.I. Kondor, J. Lafferty, Diffusion kernels on graphs and other discrete structures, in: The 19th International Conference on Machine Learning, 2002, pp. 315–322.
[34] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering method, in: International Conference on Knowledge Discovery and Data Mining, 2008, pp. 426–434.
[35] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37.
[36] J. Kubica, A. Goldenberg, P. Komarek, A. Moore, J. Schneider, A comparison of statistical and machine learning algorithms on the task of link completion, in: KDD Workshop on Link Analysis for Detecting Complex Behavior, 2003, p. 8.
[37] J. Kunegis, A. Lommatzsch, Learning spectral graph transformation for link prediction, in: International Conference on Machine Learning, 2009.
[38] J. Kunegis, E.W.D. Luca, S. Albayrak, The link prediction problem in bipartite networks, in: International Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems, 2010.
[39] S.Q. Le, T.B. Ho, T.T.H. Phan, A novel graph-based similarity measure for 2D chemical structures, Genome Informatics 14 (2004) 82–91.
[40] X. Li, H.C. Chen, Z. Zhang, J.X. Li, J.F. Nunamaker, Managing knowledge in light of its evolution process: an empirical study on citation network-based patent classification, Journal of Management Information Systems 26 (2009) 129–153.
[41] X. Li, H.C. Chen, J.X. Li, Z. Zhang, Gene function prediction with gene interaction networks: a context graph kernel approach, IEEE Transactions on Information Technology in Biomedicine 14 (2010) 119–128.
[42] T.P. Liang, Y.F. Yang, D.N. Chen, Y.C. Ku, A semantic-expansion approach to personalized knowledge recommendation, Decision Support Systems 45 (2008) 401–412.
[43] D. Liben-Nowell, J. Kleinberg, The link-prediction problem for social networks, Journal of the American Society for Information Science and Technology 58 (2007) 1019–1031.
[44] M. McPherson, L. Smith-Lovin, J.M. Cook, Birds of a feather: homophily in social networks, Annual Review of Sociology 27 (2001) 415–444.
[45] J. Newton, R. Greiner, Hierarchical probabilistic relational models for collaborative filtering, in: Workshop on Statistical Relational Learning, 21st International Conference on Machine Learning, 2004.
[46] K. Oku, S. Nakajima, J. Miyazaki, S. Uemura, Context-aware SVM for context-dependent information recommendation, in: 7th International Conference on Mobile Data Management, Washington, DC, USA, 2006, p. 109.
[47] A. Paterek, Improving regularized singular value decomposition for collaborative filtering, in: KDD Cup and Workshop, 2007, pp. 39–42.
[48] D. Pavlov, E. Manavoglu, C.L. Giles, D.M. Pennock, Collaborative filtering with maximum entropy, IEEE Intelligent Systems 19 (2004) 40–48.
[49] M.J. Pazzani, A framework for collaborative, content-based and demographic filtering, Artificial Intelligence Review 13 (1999) 393–408.
[50] G. Polcicova, P. Tino, Making sense of sparse rating data in collaborative filtering via topographic organization of user preference patterns, Neural Networks 17 (2004) 1183–1199.
[51] P.K. Reddy, M. Kitsuregawa, P. Sreekanth, S.S. Rao, A graph based approach to extract a neighborhood customer community for collaborative filtering, Lecture Notes in Computer Science 2544 (2002) 188–200.
[52] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, GroupLens: an open architecture for collaborative filtering of netnews, in: The ACM 1994 Conference on Computer Supported Cooperative Work, 1994, pp. 175–186.
[53] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Neural Information Processing Systems, 2008, pp. 1257–1264.
[54] R.R. Sarukkai, Link prediction and path analysis using Markov chains, Computer Networks 33 (2000) 377–386.
[55] B.M. Sarwar, G. Karypis, J.A. Konstan, J. Reidl, Item-based collaborative filtering recommendation algorithms, in: The International Conference on the World Wide Web, 2001, pp. 285–295.
[56] B. Scholkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, R.C. Williamson, Estimating the support of a high-dimensional distribution, in: MSR-TR-99-87, Microsoft Research, 1999.
[57] J. Scripps, P.-N. Tan, F. Chen, A.-H. Esfahanian, A matrix alignment approach for link prediction, in: International Conference on Pattern Recognition, 2008.
[58] Y. Tan, J. Wang, A support vector machine with a hybrid kernel and minimal Vapnik–Chervonenkis dimension, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 385–395.
[59] B. Taskar, P. Abbeel, D. Koller, Label and link prediction in relational data, in: IJCAI Workshop on Learning Statistical Models from Relational Data, 2003.
[60] S. Vucetic, Z. Obradovic, Collaborative filtering using a regression-based approach, Knowledge and Information Systems 7 (2005) 1–22.
[61] C. Wang, V. Satuluri, S. Parthasarathy, Local probabilistic models for link prediction, in: 7th IEEE International Conference on Data Mining, 2007, pp. 322–331.
[62] Y.W. Wang, W.H. Dai, Y.F. Yuan, Website browsing aid: a navigation graph-based recommendation system, Decision Support Systems 45 (2008) 387–400.
[63] T.F. Wu, C.J. Lin, R.C. Weng, Probability estimates for multi-class classification by pairwise coupling, Journal of Machine Learning Research 5 (2004) 975–1005.
[64] J.A. Xu, K. Araki, A SVM-based personal recommendation system for TV programs, in: The 12th International Multi-Media Modelling Conference, 2006.
[65] Y. Yajima, One-class support vector machines for recommendation tasks, in: The Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, 3918, 2006, pp. 230–239.
[66] K. Yu, W. Chu, Gaussian process models for link analysis and transfer learning, Neural Information Processing Systems, 2007.
[67] K. Yu, A. Schwaighofer, V. Tresp, X.W. Xu, H.P. Kriegel, Probabilistic memory-based collaborative filtering, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 56–69.
[68] C. Zeng, C.X. Xing, L.Z. Zhou, X.H. Zheng, Similarity measure and instance selection for collaborative filtering, International Journal of Electronics Commerce 8 (2004) 115–129.
[69] L. Zhen, G.Q. Huang, Z.H. Jiang, Recommender system based on workflow, Decision Support Systems 48 (2009) 237–245.
[70] T. Zhou, J. Ren, M. Medo, Y.C. Zhang, Bipartite network projection and personal recommendation, Physical Review E 76 (2007).
[71] J. Zhu, J. Hong, J.G. Hughes, Using Markov chains for link prediction in adaptive Web sites, in: ACM SIGWEB Hypertext, 2002.
[72] C.-N. Ziegler, S.M. McNee, J.A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: The International Conference on the World Wide Web, 2005, pp. 22–32.

Xin Li received the B. Eng. degree in automation and the M. Eng. degree in control theory and control engineering from Tsinghua University, Beijing, China, in 2000 and 2003, respectively, and the Ph.D. degree in management information systems from the University of Arizona, Tucson, in 2009. He is currently an Assistant Professor in the Department of Information Systems at the City University of Hong Kong, Hong Kong. His work has appeared in the Journal of Management Information Systems, Journal of Biomedical Informatics, Bioinformatics, Nature Nanotechnology, and Journal of the American Society for Information Science and Technology, among others. His current research interests include business intelligence & knowledge discovery, social network analysis, social media, and scientometric analysis. Dr. Li is a member of IEEE, ACM and AIS.

Hsinchun Chen received the B.S. degree in management science from the National Chiao-Tung University in Taiwan in 1981, the MBA degree from the State University of New York at Buffalo in 1985, and the Ph.D. degree in Information Systems from the New York University in 1989. He is currently the McClelland Professor of Management Information Systems at the University of Arizona, Tucson, where he is also the Director of the Artificial Intelligence Laboratory. He is author/editor of 20 books, 25 book chapters, 230 SCI journal articles, and 140 refereed conference articles. His current interests include Web computing, search engines, digital library, intelligence analysis, biomedical informatics, data/text/web mining, and knowledge management Dr. Chen is a fellow of IEEE and AAAS and a member of the ACM, INFORMS, AIS, AAAI, AMIA, and ASIS. Dr. Chen has served as a Scientific Counselor/Advisor of the National Library of Medicine (USA), Academia Sinica (Taiwan), and National Library of China (China). Dr. Chen is the Editor-in-Chief of the ACM Transactions on Management Information Systems and Springer Security Informatics Journal, and Associate Editor-in-Chief of the IEEE Intelligent Systems and serves on ten editorial boards including IEEE Transactions on Systems, Man, and Cybernetics ACM Transactions on Information Systems, Journal of the American Society for Information Science and Technology, Decision Support Systems, and International Journal on Digital Library. Dr. Chen received the IEEE Computer Society Technical Achievement Award in 2006 and the INFORMS Design Science Award in 2008.