2013 IEEE 25th International Conference on Tools with Artificial Intelligence

A Versatile Graph-based Approach to Package Recommendation

Roberto Interdonato, Salvatore Romeo, Andrea Tagarelli
DIMES, University of Calabria
87036 Arcavacata di Rende (CS), Italy
Email: {rinterdonato,sromeo,tagarelli}@dimes.unical.it

George Karypis
Department of Computer Science & Engineering,
Digital Technology Center, University of Minnesota,
Minneapolis, MN, 55455, USA
Email: karypis@cs.umn.edu

***Abstract*—An emerging trend in research on recommender systems is the design of methods capable of recommending packages instead of single items. The problem is challenging due to a variety of critical aspects, including context-based and user-provided constraints for the items constituting a package, but also the high sparsity and limited accessibility of the primary data used to solve the problem. Most existing works on the topic have focused on a specific application domain (e.g., travel package recommendation), thus often providing ad-hoc solutions that cannot be adapted to other domains. By contrast, in this paper we propose a versatile package recommendation approach that is substantially independent of the peculiarities of a particular application domain. A key aspect in our framework is the exploitation of prior knowledge on the content type models of the packages being generated that express what the users expect from the recommendation task. Packages are learned for each package model, while the recommendation stage is accomplished by performing a PageRank-style method personalized w.r.t. the target user's preferences, possibly including a limited budget. Our developed method has been tested on a TripAdvisor dataset and compared with a recently proposed method for learning composite recommendations.**

I. Introduction

Recommender systems are essential part in a variety of information-providing services that aim to satisfy their users' personalized needs. Emerging applications in e-commerce, web search, and web services integrated with social media networks are demanding for systems that are capable of producing enhanced quality recommendations which take into account the heterogeneity in the type of information to personalize and deliver to the users. As a matter of fact, recommending groups of items is the key to successfully face a number of applications, ranging from business and leisure (e.g., trip planning) to education (e.g., course combination), from finance (e.g., stock market investing) to health-care (e.g., diet planning). In all such applications the items of interest are naturally associated with different types; for example, hotels, restaurants, and points-of-interest in a travel scenario. Therefore, it is highly desirable that recommendations are provided in the form of multi-typed sets of items, or *packages*.

While the known issues in recommender systems extend to the recommendation of packages, providing suggestion lists of packages instead of single items undergoes a number of new challenges. The primary information used to drive the recommendation process still corresponds to the user-item ratings. These are characterized by high sparsity in many domains whereby items are associated with a cost (i.e., price, time) besides a value/score. The volume and quality of the primary data used to learn the packages is also negatively affected by such a high sparsity, but also by the intrinsic difficulty in satisfying different kinds of constraints, which involve compatibility and correlations among items as well as user-specified constraints (e.g., limited budget). After all, several problems for package recommendation have been shown NP-hard, as discussed in [1].

Majority of existing approaches to package recommendation have focused on a particular application domain, usually motivated by very attractive application fields such as tourism. In that case, the design of an effective recommender system has to rely on how well the specific domain challenges have been addressed, often resulting in the development of ad-hoc, hardly generalizable solutions (e.g., [2], [3], [4]). By contrast, there has also been a host of work dealing with set/package recommendation for a generic class of data (e.g., [5], [1], [6], [7]); however, while in some cases being able to express complex constraints and focusing on efficiency or optimization aspects, such studies propose overly sophisticated and rigid systems, and hence how much they could be easy-to-set and really applicable is not clear. Moreover, a common tendency in addressing the package recommendation problem is to develop solutions that are mainly based on top-k query processing (e.g., [6], [7]), combinatorial optimization (e.g., [5]), and statistical models (e.g., [2]), but surprisingly with a limited use of collaborative filtering and graph-based authority ranking techniques. While collaborative filtering should be an essential element in any information personalization task, authority-based ranking methods, such as PageRank, are ever-increasingly applied in a number of information networks, including those supporting recommender systems [8], [9], [10], [11].

We conceive the *package recommendation* problem as follows: Given a set of items of different types along with contextual information, a set of users along with their item ratings, and given prior knowledge on a set of package models of interest: learn how items can be grouped to form context-aware packages that conform to the specified models, and rank the learned package instances specifically for any target user. Our framework, named *PackRec*, consists of two stages, as sketched in Fig. 1: (i) an offline stage which is centered on

1082-3409/13 $31.00 © 2013 IEEE
DOI 10.1109/ICTAI.2013.130

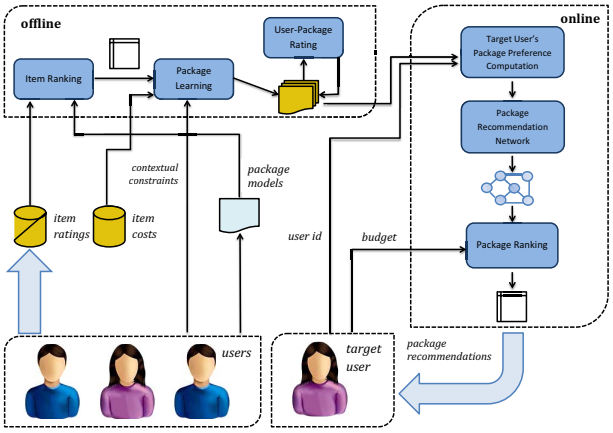

Fig. 1. Proposed package recommendation framework.

the notion of package model(s) with the objective of inducing packages for each of the specified package models, and (ii) an online stage which is in charge of recommending packages tailored to the target user's preferences and budget.

A major novelty of our proposal concerns the definition of a package recommendation framework that integrates well-established paradigms in information retrieval such as expert finding, collaborative filtering, and graph-based ranking methods: expert finding is used to estimate the relevance of items w.r.t. a package model in a collaborative fashion, user-based collaborative filtering is employed in an original odds-ratio based method that models the user's package preferences (on which the recommendation network is built), and a biased PageRank method is finally used to produce a ranked list of recommendations in the form of packages. Moreover, our approach features a certain *versatility* as it can deal with a wide range of application domains, and can work under different settings concerning the structure of the packages to be recommended and even the lack of critical information like the costs of items and/or the limited budget of users.

A preliminary experimentation on a TripAdvisor dataset has shown the recommendation ability of our approach despite the evident criticality of the selected domain. Furthermore, a comparative evaluation with a recently proposed method for composite recommendations has emphasized the superiority of our hypothesis of integration of different information retrieval techniques against a strategy based on an instance-optimal cost/value-item-driven knapsack.

II. Definitions and Notation

We are given a set of users $U = \{u_1, \ldots, u_m\}$ and a set of items $I = \{e_1, \ldots, e_n\}$. Users' rating information is stored into a matrix $\mathbf{R}_I$, where each entry $R_{ue} \in \mathbb{R}^*$ corresponds to the rating given by user u to item e (zero in case of no rating). Moreover, each item might be associated with a real value denoting a cost; similarly, each user could specify a budget, in relation to a given context and time. Optionally, there may be temporal information about when users have rated/reviewed the items; in this case, under the usual assumption that more recent ratings better match the user's current preferences, the importance of a rating would be decreased inversely proportional to its timestamp, for which purpose we will opt for a logarithmic function: for an item e rated by user u with value v at time t, the actual R_{ue} will be set as $R_{ue} = v + v\log(1/(1+\Delta t))$, where Δt is the difference between the current timestamp and t.

TABLE I
Main Notations Used in This Paper

(in) var.	*description*	*(out) var.*	*description*
u, U	user, set of users	p	package
e, I	item, set of items	P	set of packages
$\mathbf{R}_I$	user-item rating matrix	$\mathbf{R}_P$	user-package rating matrix
R_{ue}	rating of item e by user u	R_{ui}	rating of package p_i by user u
$\mathcal{I}$	item type	*primary param.*	*description*
$\boldsymbol{\mathcal{I}}$	set of item types	α	damping factor in Step 5
$\mathcal{P}$	package model	$Usim$	user similarity thr. in Step 4
$\boldsymbol{\mathcal{P}}$	set of package models	*secondary param.*	*description*
$\mathcal{C}$	set of contextual constraints	$\omega_{u\mathcal{I}}, \omega_{u\mathcal{P}}$	weights in Step 3
$\mathcal{S}$	set of constants	λ_u, λ_{ps}	smoothing factors in Steps 1 and 3

Each item is associated to one type $\mathcal{I}$ from a predefined set of item types $\boldsymbol{\mathcal{I}}$. We will use $\mathcal{I}_e$ to indicate the type associated to item e, and U_e to indicate the set of users related to e.

Constraints might be specified to allow for checking whether an item satisfies certain contextual conditions (e.g., same location for travel items). Therefore, assuming there can be recognized type-independent yet relevant attributes of the items (e.g., location), a set of predicate symbols $\mathcal{C}$ and a set of constants $\mathcal{S}$ are defined that correspond to the sets of attribute names and attribute values, respectively. An atom is an expression of the form $c(e, s)$ with boolean truth values, where c is a predicate symbol and s is a constant; for example, $location(e,''Rome'')$ evaluates to true if e (hotel, restaurant, or any type of attraction) is located in Rome.

Upon the basic notion of item, we define a *package* as a group of items which might be of different types. Specifically, a package p can be seen as a subset of I that conforms to a predefined *package model* $\mathcal{P}$. We assume the existence of a set of package models denoted as $\boldsymbol{\mathcal{P}}$, where each package model is a multiset over $\boldsymbol{\mathcal{I}}$, i.e., $\mathcal{P} = \langle \mathcal{I}(\mathcal{P}) \subseteq \mathcal{I}, f_{\mathcal{P}} \rangle$ where $\mathcal{I}(\mathcal{P})$ denotes the set of *valid item types* and $f_{\mathcal{P}} : \mathcal{I}(\mathcal{P}) \rightarrow \mathbb{N}^+$ indicates the number of repetitions for each item type in $\mathcal{P}$. Table I summarizes main notations that will be used throughout this paper.

III. Package Recommendation Framework

The proposed package recommendation framework (Fig. 1) takes an input set of users, items, and corresponding ratings along with knowledge on the types of items, desired package models and contextual constraints, and performs the following main steps: offline ranking of the items, learning of the packages, computation of the user-package ratings, and online ranking of the packages tailored to any target user. Each of these steps will be described in the next sections.

A. Step 1: Collaborative, package-model-aware item ranking

The first step is in charge of identifying the best candidate items to form packages for each of the known models. Given

a package model $\mathcal{P}$, the relevance of each item e having type $\mathcal{I}_e$ valid for $\mathcal{P}$ is computed using a probabilistic model, which takes into account three main criteria: (i) the likelihood of e given $\mathcal{P}$, (ii) the likelihood of $\mathcal{I}_e$ given $\mathcal{P}$, and (iii) the a-priori likelihood of e. We now elaborate on each of these terms.

The likelihood of e given $\mathcal{P}$ is expressed by an expert finding model, where $\mathcal{P}$ is assumed to be conditionally independent to e given u:

$$\Pr(e|\mathcal{P}) = \sum_{u \in U_e} \Pr(e|u) \Pr(u|\mathcal{P}) \tag{1}$$

Probability $\Pr(e|u)$ measures the relevance of item e for user u and is estimated as

$$\Pr(e|u) = \frac{R_{ue}}{\sum_{e' \in I, \mathcal{I}(e') = \mathcal{I}_e} R_{ue'}} \tag{2}$$

Probability $\Pr(u|\mathcal{P})$ can be rewritten as $\Pr(\mathcal{P}|u) \Pr(u)$. $\Pr(u)$ expresses the strength of a user based on her/his degree of activity of rating, which is simply estimated as

$$\Pr(u) = \frac{\sum_{e \in I} \delta_u(e)}{\sum_{u' \in U} \sum_{e \in I} \delta_{u'}(e)} \tag{3}$$

where $\delta_u(e) = 1$ if u has rated item e, and zero otherwise. Probability $\Pr(\mathcal{P}|u)$ is determined using a zeroth-order Markov model:

$$\Pr(\mathcal{P}|u) = \Pr(\mathcal{P}|\theta_u) = \prod_{\mathcal{I} \in \mathcal{P}} \Pr(\mathcal{I}|\theta_u) \tag{4}$$

where $\Pr(\mathcal{I}|\theta_u)$ can be estimated as: $\Pr(\mathcal{I}|\theta_u) = (1 - \lambda_u) \Pr(\mathcal{I}|u) + \lambda_u \Pr(\mathcal{I}|U)$, with smoothing factor $\lambda_u \in [0, 1]$ as an input parameter (set to 0.1 by default).

The likelihood of $\mathcal{I}_e$ given $\mathcal{P}$, i.e., $\Pr(\mathcal{I}_e|\mathcal{P})$, is estimated by linear interpolated smoothing (a.k.a. Jelinek-Mercer):

$$\begin{aligned} \Pr(\mathcal{I}_e|\mathcal{P}) &\propto \Pr(\mathcal{I}_e|\theta_{\mathcal{P}}) = \\ &= (1 - \lambda_{\mathcal{P}}) \Pr(\mathcal{I}_e|\mathcal{P}) + \lambda_{\mathcal{P}} \Pr(\mathcal{I}_e|\boldsymbol{\mathcal{P}}) \end{aligned} \tag{5}$$

where $\Pr(\mathcal{I}_e|\mathcal{P}), \Pr(\mathcal{I}_e|\boldsymbol{\mathcal{P}})$ are maximum likelihood estimators, and $\lambda_{\mathcal{P}} \in [0, 1]$ is a smoothing factor; recall that, for a generic element x, a multi-set $\mathcal{X}$, and a set of multi-sets $\boldsymbol{\mathcal{X}}$, the maximum likelihood estimators are determined as $f_{\mathcal{X}}(x)/|\mathcal{X}|$ and $\sum_{\mathcal{X} \in \boldsymbol{\mathcal{X}}} f_{\mathcal{X}}(x) / \sum_{\mathcal{X} \in \boldsymbol{\mathcal{X}}} |\mathcal{X}|$, respectively. The smoothing factor $\lambda_{\mathcal{P}}$ is set based on the following hypothesis: the degree of smoothing should be higher for package models that are less frequent and more complex; formally, it is defined as $\lambda_{\mathcal{P}} = \frac{|\mathcal{P}|}{|\mathcal{P}| + mc(\mathcal{P}, \boldsymbol{\mathcal{P}})}$, where $mc(\mathcal{P}, \boldsymbol{\mathcal{P}})$ expresses the relative frequency of *containment* of model $\mathcal{P}$ in $\boldsymbol{\mathcal{P}}$, i.e., the fraction of package models of which $\mathcal{P}$ is subset.

Finally, prior $\Pr(e)$ is expressed analogously to (3):

$$\Pr(e) = \frac{\sum_{u \in U} \delta_u(e)}{\sum_{e' \in I} \sum_{u \in U} \delta_u(e')} \tag{6}$$

Overall, the first step is accomplished by iteratively performing the following set of equations ($\forall e \in I \; s.t. \; \mathcal{I}_e \in \mathcal{I}(\mathcal{P}), \forall \mathcal{P} \in \boldsymbol{\mathcal{P}}$) which determines a ranking of all items for the selected package model:

$$rank_{\mathcal{P}}(e) = [\Pr(e|\mathcal{P}) + \Pr(\mathcal{I}_e|\mathcal{P})] \Pr(e) \tag{7}$$

Intuitively, given $\mathcal{P}$, the importance of e according to (7) relies on the relevance of the item as a candidate to constitute $\mathcal{P}$ ($\Pr(e|\mathcal{P})$) as well as on the relevance of the item's type w.r.t. $\mathcal{P}$ ($\Pr(\mathcal{I}_e|\mathcal{P})$). Yet, $\Pr(e)$ accounts for the popularity of e independently of the generation of packages, and hence acts to penalize items that are less rated by the users.

B. Step 2: Context-driven package learning

The item rankings computed at the previous step are used to construct a set of package instances for all package models in $\boldsymbol{\mathcal{P}}$. Besides the type compatibility to ensure for the items w.r.t. any given package model $\mathcal{P}$, contextual constraints at the instance level might also be satisfied. Therefore, predefined contextual predicates could be here applied to extract a subset of $\mathcal{P}$-compatible items, for each of the $|\mathcal{P}|$ item types that conform to $\mathcal{P}$. Given a DNF formula $\mathcal{A}$ over a subset $C \subseteq \mathcal{C}$ of predicates and a subset $S \subseteq \mathcal{S}$ of constants, a set of items $I_{\mathcal{I}}^{\mathcal{A}}$ is derived for each item type $\mathcal{I} \in \mathcal{I}(\mathcal{P})$ such that $I_{\mathcal{I}}^{\mathcal{A}}$ contains the items of type $\mathcal{I}$ that also satisfies $\mathcal{A}$.

We automatically select the top-ranked items for the various types valid for a given package model by pipelining *Pareto-frontier* computations over the $I_{\mathcal{I}}^{\mathcal{A}}$. For each $I_{\mathcal{I}}^{\mathcal{A}}$, alternatives among the items are evaluated according to their normalized cost/normalized score ratio, where the item scores are computed by (7); i.e., upon an ordering of the items by increasing cost/value ratios, the next item with smallest rank is added to the Pareto-frontier if it is not dominated by any item already in the Pareto-frontier. To check whether a candidate item is dominated by any alternative in the Pareto-frontier, it will suffice to find the most expensive alternative which is still cheaper than the candidate item, since it does not dominate the candidate item, neither can any other alternative in the Pareto-frontier. It should be however noted that for some types of items, either the cost information is not applicable or the value (score) dimension should weight more than the cost of items; consequently, a cost/value based strategy for the selection of top items might not necessarily lead to the best choice in terms of item coverage. For this purpose, we also used a cost-free selection strategy such that an item will be included in the top-ranked list for type $\mathcal{I}$ if its score has a percentage decrease from the top-1 item which is not greater than the average of the percentage decreases of all items from the top-1 item.

The set of packages for each model $\mathcal{P}$ is finally computed as the result of the Cartesian product of the $|\mathcal{P}|$ sets of selected type-specific items for $\mathcal{P}$. Each package p is provided in the form of a set of items.

C. Step 3: Package rating

Once a set P of packages for all package models has been learned, user ratings for the packages need to be computed. Such ratings can certainly be inferred from the ratings of the items that belong to a package: however, the usual sparsity of the user-item matrix is expected to be further exacerbated when selecting the small bunch of items that constitutes any given package. In order to alleviate this issue, we introduce a smoothing scheme that refines the actual R_{ue}, with $e \in p$:

$(1-\lambda_{ps})R_{ue}+\lambda_{ps}\overline{R_e}$, where $\overline{R_e}$ denotes the average rating of e over all users, with λ_{ps} set to 0.1 by default.

Besides users' ratings of the items constituting a package, other information might be taken into account when computing the rating of a user for a package. A user may want to differentiate among the types of items constituting a package: given user u and package p, this can be expressed by a set of weights $\omega_{u\mathcal{I}} \in [0,1]$, with $\mathcal{I} \in \boldsymbol{\mathcal{I}}$ each of which might act as a damping (or even as nullifying) factor for a particular item type; by default, such weights are set to 1 meaning that all constituent items are fully considered when rating the package. Moreover, a user may specify prior preferences over package models, based on her/his individual exogenous source of information: this is expressed by a coefficient $\omega_{u\mathcal{P}} \in [0,1]$ (set to 1 by default), for each $\mathcal{P} \in \boldsymbol{\mathcal{P}}$.

The above types of information are combined to define a *package rating* function *p-score* : $U \times P \rightarrow \mathbb{R}^*$, which is computed for any given user u and package p as:

$$\textit{p-score}(u,p) = \omega_{u\mathcal{P}} \frac{\sum_{e \in p} \omega_{u\mathcal{I}_e}[(1-\lambda_{ps})R_{ue} + \lambda_{ps}\overline{R_e}]}{\sum_{e \in p} \omega_{u\mathcal{I}_e}} \quad (8)$$

A *user-package rating matrix*, denoted as $\mathbf{R}_P$, is derived by storing the computed *p-score*(u,p) values, for all $u \in U$ and $p \in P$. Hereinafter we will use notation R_{ui} to denote the rating given by user u to package p_i.

D. Step 4: Target user's package recommendation network

A *package recommendation network* will be designed for any given target user, and used to produce a ranking (probability distribution) over the set of packages obtained from the offline stage, thus providing recommendations to the target user. The network is modeled as a directed graph in which vertices are the learned packages and edges are drawn according to a function that models the user's preference on pairs of packages. The role of this preference function is to define the transition probabilities in the package ranking model discussed later. The sign of the preference function will determine the orientation of the edge, while a pair of reciprocal edges will be drawn in case of no preference expressed for any two packages. The strength of the connection between two packages is set to be proportional to the value of the preference function. Formally, the package recommendation network for the target user u is defined as $\mathcal{G}_u = \langle \mathcal{V}, \mathcal{E}, w_u \rangle$, with set of vertices $\mathcal{V} = P$, set of edges $\mathcal{E} = \{(p_i,p_j) | p_i,p_j \in P \wedge \pi_u(p_j,p_i) \geq 0\}$, and edge weighting function $w_u : \mathcal{E} \rightarrow \mathbb{R}^+$ such that $w_u(i,j) = e^{\pi_u(p_j,p_i)}$.

Package preference function: Given a user u, our objective is to model a function of the form $\pi_u : P \times P \rightarrow \mathbb{R}$, with $\pi_u(p_i,p_j) > 0$ when package p_i is considered as more preferable to package p_j for user u, and vice versa, whereas $\pi_u(p_i,p_j) = 0$ means that there is no preference between the two packages. We require that $\pi_u(p_i,p_i) = 0$, for all $p_i \in P$ and $\pi_u(p_i,p_j) = -\pi_u(p_j,p_i)$, for all $p_i,p_j \in P$. Moreover, to adequately determine the strength of preference, we are interested in modeling the target user's package preferences in a collaborative setting.

Let us denote with U_{ij} the set of users that have rated p_i or p_j, or both. We define in terms of *odds ratio* the strength of association of two random variables: the one expressing the condition "users in U_{ij} are similar to u" and the other one expressing the event "p_i is rated higher than p_j". Hence, an odds ratio greater than (resp. lower than) 1 indicates that the event of rating p_i higher than p_j is more likely to occur in the group of users similar (resp. not similar) to u, while an odds ratio equal to 1 indicates that whether or not users are similar to u is irrelevant to prefer p_i over p_j.

To determine user similarity, we can resort to various similarity measures used in collaborative filtering [10]. The extent to which two users are considered as similar to each other can be controlled by a minimum-similarity threshold, henceforth denoted as $Usim$, which can be experimentally varied.

The collaborative-based preference odds ratio computed for each pair p_i,p_j is then combined with the variation in u's ratings of p_i,p_j to model the preference function.

$$\pi_u(p_i,p_j) = (R_{ui} - R_{uj})(OR_{u,ij})^{1-2\chi_{u,i,j,OR}} \quad (9)$$

where $\chi_{u,i,j,OR}$ is the indicator function for the event "$(R_{ui} - R_{uj}) < 0$", and

$$OR_{u,ij} = \frac{N_u^{>}}{N_u^{<}} \cdot \frac{notN_u^{<}}{notN_u^{>}} \quad (10)$$

with $N_u^{>}$ (resp. $notN_u^{>}$) indicating the add-one smoothed number of users similar (resp. not similar) to u that have rated p_i strictly higher than p_j, $N_u^{<}$ (resp. $notN_u^{<}$) indicating the add-one smoothed number of users similar (resp. not similar) to u that have rated p_i strictly lower than p_j.

It can be noted that the two terms in (9) play a different role in modeling the preference function: while the "versus" of preference is determined by $(R_{ui} - R_{uj})$, the odds ratio acts as a strengthening (resp. damping) factor if there is a concordance (resp. discordance) in sign between the difference of u's ratings and the logarithm of the odds ratio. Note also that, as required, $\pi_u(p_i,p_j) = -\pi_u(p_j,p_i)$ since it holds that $OR_{u,ij} = OR_{u,ji}^{-1}$ and $\chi_{u,i,j,OR} = 1 - \chi_{u,j,i,OR}$, which ensure that the overall odds ratio term is identical in both $\pi_u(p_i,p_j)$ and $\pi_u(p_j,p_i)$.

E. Step 5: Package ranking

We develop a PageRank-style method for ranking the set of packages specifically for each target user. PageRank-style methods have been already successfully applied to item recommendation problems [8], [9], [10], [11]. A major motivation is that the underlying Markov chain model is effective to face the usual lack of rating or preference information that characterize many users. In other terms, if preferences between packages p_i,p_j have been expressed by some users, and preferences between packages p_i,p_k have been expressed by other users, the preferences regarding p_j,p_k are not known and hence can be inferred through an iterated random walk. In our package recommendation setting, an intuitive interpretation of the classic PageRank idea is that the importance of a package (i.e.,

the likelihood of being preferred to other packages) both relies on and influences the importance of neighboring packages in the network $\mathcal{G}_u$. This is captured by the following equation that computes the ranking score r_i for a given package p_i:

$$r_i = \sum_{j \in In(i)} \frac{w_u(j,i)}{\sum_{h \in Out(j)} w_u(j,h)} r_j \quad (11)$$

where, for any vertex i, $In(i)$ and $Out(i)$ denote the in-neighbor and out-neighbor sets, respectively. The above equation can be written in the equivalent matrix notation as $\mathbf{r} = \mathbf{S}^{\mathrm{T}}\mathbf{r}$, where $\mathbf{S}$ is the *package connectivity* stochastic matrix defined as $\mathbf{S} = {\mathbf{D}_{out}}^{-1}\mathbf{W} + \mathbf{a}\mathbf{e}^{\mathrm{T}}/|\mathcal{V}|$, such that $\mathbf{W}$ is the weighted adjacency matrix of $\mathcal{G}_u$, $\mathbf{D}_{out} = diag(\mathbf{W}\mathbf{e})$ is the diagonal matrix storing the (weighted) out-degrees with $\mathbf{e}$ denoting a $|\mathcal{V}|$-dimensional column vector of ones, and $\mathbf{a}$ is the dangling-vertex vector such that $a_i = 1$ if vertex i has zero out-degree (i.e., package p_i is not rated), and 0 otherwise. To ensure the convergence of the Markov chain with $\mathbf{S}$ to a stationary distribution, the usual primitivity adjustment is introduced as a convex combination of $\mathbf{S}$ with another stochastic matrix defined as $\mathbf{B} = \mathbf{v}\mathbf{v}^{\mathrm{T}}/|\mathcal{B}|$. Vector $\mathbf{v}$ and set $\mathcal{B}$ are equal by default to $\mathbf{e}$ and $\mathcal{V}$, respectively, but $\mathbf{v}$ can be replaced with any vector whose non-negative components sum up to 1 and that can be used to personalize the PageRank to boost a specific subset of vertices (base-set $\mathcal{B}$). $\mathbf{B}$ is also known as teleportation matrix, since the random walker can decide not to follow the link structure by selecting a vertex with relevance $1/|\mathcal{B}|$. A parameter α between 0 and 1 (commonly set to 0.85) controls the proportion of the random walk based on the link structure as opposed to teleporting: $\mathbf{r} = \alpha\mathbf{S}^{\mathrm{T}}\mathbf{r} + (1-\alpha)\mathbf{v}/|\mathcal{B}|$.

We define a *cost-sensitive personalization* of our PageRank-based method, according to the following two criteria: (i) $\mathcal{B}$ corresponds to the subset of packages in P having the same model as the packages that have been rated by u, and (ii) each of the selected packages in $\mathcal{B}$ must have a cost (computed over its constituting items) not greater than a specified budget b_u.

IV. Related Work

In the last few years recommendation of sets/packages of items has been studied from various perspectives and with different levels of expressiveness. [12] proposes a general decision support system for the definition of composite alternative recommendations. While different user-specified requirements are taken into account, the recommendation process is highly interactive, as it utilizes the target user's feedback to feed a preference learner module, and hence it might continue iteratively until the user is satisfied with one of the alternatives. [6] approaches the problem from a top-k query processing perspective, by introducing the class of entity package finder query. Such queries are used to identify the top-k tuples of entities, according to the relevance of each entity w.r.t. a given set of keywords. Like our approach, associations among the entities (i.e., item types) are known in advance; however, entity package finder queries cannot directly handle user-specified constraints such as budget to control the identification of most relevant packages and, in general, they do not consider a collaborative-based support to provide a personalized recommendation to a target user. In [7], complex yet customized recommendations are defined declaratively as a high-level workflow over relational data, in which traditional and enhanced relational algebra operators are used to specify user-requirements and to generate virtual nested relations. The approach in [7] is designed to maximize flexibility in recommendation, however at the cost of higher complexity of the recommendation engine and difficulty to control contextual and cost requirements.

Unlike our work, the approaches in [4], [3] exploit geo-temporal and multimedia data to provide recommendations about popular touristic places. Both studies however do not take item/package rating into consideration, thus they are limited to provide composite recommendations for a set of user-specified temporal constraints. The latter differences w.r.t. our work are also present in [13], which studies the general problem of set-based queries with aggregation constraints.

In [2], travel packages are generated through the TAST (tourist-area-season topic) model, based on latent topic distributions of tourist-season pairs. Such topic distributions are used to find seasonal nearest neighbors for each tourist, and collaborative filtering based ranking is employed to personalize suggestions. An extension of the TAST model, named TRAST (tourist-relation-area-season topic) is also defined to model the tourist relationships in a travel group. The approach in [2] can provide recommendations only for users who have traveled at least once in the existing travel records, i.e., users who have rated at least one package, while our approach can even handle target user's partial ratings about the items constituting the candidate packages to recommend. While in our model a package might contain different item types, each one with arbitrary multiplicity, in [2] a package is substantially an array of landscapes. Also, the TAST/TRAST models are in principle applicable to other scenarios but only provided that certain assumptions hold at the basis of the topic model.

The composite recommendation system proposed in [5] is centered on a domain-independent approach that extends the knapsack problem. It assumes the availability of multiple recommender systems to score the items for a specific user and of an external information source that provides the items' costs. Given a cost budget and an integer k, the method computes top-k variable-length packages to recommend. Two approximation algorithms are developed, an instance-optimal, pseudo-polynomial algorithm and a greedy algorithm. Unlike our work, packages are learned regardless of the specific type and compatibility constraints of the constituting items; moreover, the knapsack-like approach adopted in [5] is by nature not particularly tailored to the various target user's preferences, which are in fact simply summarized in the specification of the budget constraint and whose understanding cannot easily be refined in a collaborative fashion.

Nevertheless, we involved the approach by [5] in a comparative evaluation with our PackRec, not only because the two works are both domain-independent and do not rely on

TABLE II
PACKAGE MODELS AND STATISTICS ABOUT THE LEARNED PACKAGES.

Group	Location	# Ratings per Location	# Learned Packages	Percentage of packages covered by each budget range	Package Models
1	Amsterdam	58,746	126	3.97%, 28.57%, 54.76%, 84.92%	$\mathcal{P}_1$: Hotel, Restaurant, Museums, HistoricSites, ArchitecturalBuildings
	Buenos Aires	56,926	64	15.63%, 18.75%, 51.56%, 56.25%, 81.25%	$\mathcal{P}_2$: Hotel, Restaurant, Landmarks/PointsofInterest, Parks, Entertainment
	Istanbul	59,538	64	15.63%, 60.94%	$\mathcal{P}_3$: B&B, Restaurant, Performances, Entertainment
					$\mathcal{P}_4$: Hotel, Restaurant, HistoricSites, Civic/ConventionCenters
2	Barcelona	73,389	68	4.41%, 7.35%, 19.12%, 54.41%, 82.35%	$\mathcal{P}_5$: Hotel, Restaurant, Museums, HistoricSites, ReligiousSites
	Chicago	60,852	64	75.00%, 87.50%	$\mathcal{P}_6$: Hotel, Restaurant, HistoricSites, Gardens
	Honolulu	38,049	80	11.25%, 18.75%, 45.00%, 62.50%	$\mathcal{P}_7$: B&B, Restaurant, Museums, Theaters
	San Francisco	69,041	64	62.50%, 87.50%, 93.75%	$\mathcal{P}_8$: Hotel, Restaurant, HistoricSites
	Sydney	39,276	68	5.88%, 14.71%, 75.00%, 98.53%	
	WashingtonDC	46,403	64	14.06%, 25.00%, 76.56%, 95.31%	
3	Edinburgh	52,382	112	21.43%, 57.14%, 77.68%	$\mathcal{P}_9$: Hotel, Restaurant, Museums, ArchitecturalBuildings, ReligiousSites
	Rome	28,428	104	15.38%, 23.08%, 30.77%, 65.38%, 82.69%, 88.46%	$\mathcal{P}_{10}$: Hotel, Restaurant, Parks, Entertainment
	Venice	51,873	64	17.19%, 37.50%, 50.00%, 70.31%, 79.69%	$\mathcal{P}_{11}$: B&B, Restaurant, Performances, Entertainment
					$\mathcal{P}_{12}$: Hotel, Restaurant, ArchitecturalBuildings, ReligiousSites
4	Niagara Falls	17,843	96	3.13%, 11.46%, 40.63%, 90.63%	$\mathcal{P}_{13}$: Hotel, Restaurant, Theaters
	Playa del Carmen	28,629	88	5.68%, 12.50%, 15.91%, 18.18%, 72.73%, 86.36%	$\mathcal{P}_{14}$: Hotel, Restaurant, Entertainment, SportsCamps/Clinics
	Sharm El Sheikh	21,431	64	6.25%, 21.88%, 40.63%, 59.38%, 75.00%	$\mathcal{P}_{15}$: B&B, Restaurant, Entertainment, SportsCamps/Clinics
					$\mathcal{P}_{16}$: Hotel, Restaurant, Theaters, Parks

a text processing via language modeling (like [2] does), but also because this evaluation allowed us to stress our PackRec under a different setting in which the schema of the learned packages can be highly varying and is not user-provided.

V. EXPERIMENTAL EVALUATION

A. Data and evaluation methodology

To assess our approach we chose to focus on the travel planning domain, given the increasing interest it has produced as a major application for package recommendation tasks. We used the popular TripAdvisor.com data as case in point for our evaluation. During April 2013, we crawled information about hotels, restaurants, and all available types of attractions along with the associated users' ratings, starting from the Top-Destinations section of the website[1]. This resulted in $48,131$ hotels, $19,802$ B&Bs, $159,716$ restaurants, and $21,661$ attractions classified in 133 categories, with a total of $249,310$ items, and $12,622,091$ ratings made by $4,004,926$ users over $230,814$ items. However, the very high sparsity of the rating matrix (above 99%) prompted us to restrict our selection to a subset of locations in the attempt of reaching a good tradeoff between salience of the venue in TripAdvisor (in terms of user popularity), diversity in terms of the attractions a venue usually offers (i.e., diversity in the set of item types), and suitability of the locations to perform queries related to different travel topics (i.e., nature, business, historical sites, etc.). As a result, we selected 15 locations which are shown in Table II.

To assess the proposed and competing methods, we assigned a reference ranking score to each package p, which takes into account the TripAdvisor-supplied rankings and costs of the constituting items as well as a user-provided budget (b):

$$rank(p,b) = \text{avg}_{\mathcal{I}\in\mathcal{P}}\left(\frac{\sum_{e:\mathcal{I}_e=\mathcal{I}} rank(e)}{\sum_{i=1..m(\mathcal{I}_e)} i}\right)\frac{1}{(cost(p)-b)+1}$$

where $rank(e)$ is the rank of item e w.r.t. the item-type specific ranking provided by TripAdvisor, $cost(p)$ is the total cost of package p calculated over its items' costs, b equals the budget specified by the target user (i.e., $b = b_u$), and $|p|$ denotes the number of item-types (with duplicates) for the package model of p. Note that the formula penalizes out-of-budget packages.

We used four assessment criteria that are standard in ranking tasks, namely *mean average precision* (MAP), *Kendall rank correlation coefficient*, *normalized discounted cumulative gain* (nDCG), and *Fagin's intersection metric*. For each of them, higher scores correspond to better ranking evaluation.

MAP is the mean value of the average precisions computed for a set of queries. The average precision for a single query is calculated as $AP = \frac{\sum_{n=1}^{k} P@n \cdot rel(n)}{|R|}$, where $P@n$ is precision at step n (i.e., fraction of the top-n retrieved results that are relevant for the given query) and $|R|$ is the number of relevant candidates. In our setting, the personalized ranking produced for a given user is considered as a query, and the number of relevant and retrieved candidates is obtained by taking into account the top-k-ranked packages.

Let us denote with $\mathcal{L}^*$ and $\mathcal{L}$ the reference ranking and the ranking produced by an algorithm, respectively, and with $\mathcal{L}(i)$ the ranking value associated to the package ranked in position i. The Kendall rank correlation coefficient evaluates the similarity between two rankings, expressed as sets of ordered pairs, based on the number of inversions of package pairs which would be needed to transform one ranking into the other. It is computed as: $Kendall(\mathcal{L}^*, \mathcal{L}) = 1 - \frac{2\Delta(\mathcal{P}(\mathcal{L}^*),\mathcal{P}(\mathcal{L}))}{N(N-1)}$ where $N = |\mathcal{L}| = |\mathcal{L}^*|$ and $\Delta(\mathcal{P}(\mathcal{L}^*), \mathcal{P}(\mathcal{L}))$ is the number of unshared package pairs between the two lists.

nDCG [14] measures the usefulness (gain) of a package based on its relevance and position in a list. Formally, nDCG is the ratio between the discounted cumulative gain to its ideal (reference) counterpart taking into account the top-k-ranked packages in two lists: $nDCG^{(k)} = \frac{DCG^{(k)}}{IDCG^{(k)}}$, such that $DCG^{(k)} = \mathcal{L}^*[\arg\mathcal{L}(1)] + \sum_{i=2}^{k} \frac{\mathcal{L}^*[\arg\mathcal{L}(i)]}{\log_2(i+1)}$, where symbol $\arg\mathcal{L}(i)$ is used to denote the package number of the package ranked at position i in the algorithm's ranking (i.e., $\mathcal{L}^*[\arg\mathcal{L}(i)]$ is the reference ranking value for that package), and $IDCG^{(k)} = \mathcal{L}^*(1) + \sum_{i=2}^{k} \frac{\mathcal{L}^*(i)}{\log_2(i+1)}$.

Fagin's intersection metric [15] is used to compare partial

[1] http://www.tripadvisor.com/TravelersChoice-Destinations

rankings, where elements in one list may not be present in the other. It applies to any two top-k lists: $F(\mathcal{L}^*, \mathcal{L}, k) = \frac{1}{k}\sum_{i=1}^{k} \frac{|\mathcal{L}^*_{:i} \cap \mathcal{L}_{:i}|}{i}$, where $\mathcal{L}^*_{:i}$, $\mathcal{L}_{:i}$ denote the sets of packages from the 1st to the ith position in the respective rankings.

B. Experimental settings

PackRec parameters were setup using the default values as declared in their definitions; moreover, cosine similarity and $Usim$ set to 0.6 were used for the user neighborhood computations (cf. Sect. III-D). We however undertook a preliminary investigation on how the PackRec performance was influenced by λ_{ps} (cf. Sect. III-C) and by $Usim$: in summary, we observed that a value for λ_{ps} close to zero (e.g., the default 0.1) was enough to alleviate the sparsity issue but also to avoid a loss of discrimination between the individual users' ratings in scoring the packages; similar neighborhood sizes were observed when setting $Usim$ to 0.6÷0.8, and even down to 0.5 in a few locations, while lower (resp. higher) values would lead to a nearly total (resp. null) coverage of the users.

We devised two evaluation stages, the first focused on the assessment of our PackRec performance under different settings, and the second devoted to a comparative evaluation with the instance-optimal algorithm in [5], hereinafter referred to as CompositeRec, as anticipated at the end of Sect. IV.

In the first stage, the selected locations were grouped into four categories according to their attractions. As shown in Table II, we defined a set of four package models based on the attraction types available in the corresponding locations and with the attempt of configuring four types of trip, namely cultural, family, business, and fun trip. We specified the input budgets for each location trying to simulate different price ranges, according to the following strategy: we calculated the total cost of each package learned contextually to a given location, then we analyzed the package costs in increasing order, fixing a budget threshold when the next cost was a certain percentage (set to 20%) higher than the lower cost in the current range. Note that the setting corresponding to the highest budget (i.e., budget set equal to the maximum package cost for the location) will correspond to no budget-driven personalization of the ranking.

In the second stage, since CompositeRec requires a number of recommender systems in input, we exploited a state-of-the-art method, called ItemRank [11], to generate personalized item rankings for the five most active users for each location. Since the package models of the packages returned by CompositeRec cannot be known in advance, we carried out CompositeRec to recommend the top-10 packages for each of the selected users per location, by setting the budget per location as equal to the corresponding highest budget used by our PackRec in the first stage. Then, the models of the packages produced by CompositeRec were used to drive the package learning step of PackRec. For the comparative evaluation of the recommendations, we needed to generate a different reference ranking for each user, containing both the packages returned by CompositeRec and by PackRec, constrained to the same budget per location.

TABLE III
AVERAGE PERFORMANCES OF PACKREC.

Location	Fagin		Kendall		nDCG		MAP
	avg	*max*	*avg*	*max*	*avg*	*max*	
upper-quartile users							
Amsterdam	.228	.393	.432	.557	.503	.625	.193
BuenosAires	.024	.218	.165	.641	.173	.945	.038
Istanbul	.175	.503	.417	.574	.440	.783	.143
group avg	.129	.342	.315	.597	.348	.796	.116
Barcelona	.112	.246	.271	.435	.317	.473	.123
Chicago	.153	.602	.223	.689	.416	.871	.152
Honolulu	.125	.398	.296	.422	.385	.707	.113
SanFrancisco	.115	.609	.172	.617	.464	.822	.123
Sydney	.184	.646	.156	.419	.479	.836	.160
WashingtonDC	.128	.402	.210	.447	.437	.779	.112
group avg	.135	.462	.225	.485	.411	.727	.129
Edinburgh	.170	.552	.349	.608	.422	.924	.205
Rome	.165	.497	.426	.646	.308	.628	.120
Venice	.096	.302	.262	.399	.304	.509	.071
group avg	.142	.441	.350	.550	.334	.655	.123
NiagaraFalls	.117	.231	.182	.327	.361	.518	.099
PlayadelCarmen	.185	.455	.192	.549	.428	.768	.182
SharmElSheikh	.103	.366	.219	.440	.379	.748	.102
group avg	.139	.363	.198	.451	.393	.692	.132
lower-quartile users							
Amsterdam	.191	.252	.443	.525	.474	.609	.161
BuenosAires	.099	.389	.324	.555	.221	.595	.069
Istanbul	.148	.336	.466	.604	.383	.571	.122
group avg	.142	.329	.397	.555	.346	.595	.114
Barcelona	.102	.120	.272	.411	.267	.318	.123
Chicago	.079	.261	.155	.531	.356	.656	.108
Honolulu	.147	.351	.377	.532	.448	.796	.162
SanFrancisco	.146	.649	.253	.487	.545	.784	.130
Sydney	.053	.098	.305	.522	.341	.496	.058
WashingtonDC	.154	.373	.274	.533	.444	.741	.125
group avg	.114	.293	.281	.498	.393	.613	.118
Edinburgh	.180	.520	.439	.664	.501	.869	.204
Rome	.220	.522	.495	.651	.400	.646	.192
Venice	.065	.233	.241	.391	.256	.543	.051
group avg	.155	.419	.393	.562	.373	.662	.149
NiagaraFalls	.156	.787	.481	.592	.766	.925	.161
PlayadelCarmen	.123	.239	.321	.594	.348	.512	.102
SharmElSheikh	.166	.454	.341	.574	.489	.778	.180
group avg	.148	.463	.372	.587	.511	.715	.148

C. Results

Table III reports on performance results obtained by PackRec over the various locations; parameter k in Fagin, nDCG and MAP was set to cover 10% of the package set. For each location and assessment criterion, results correspond to averages over the different budgets selected for the location. Moreover, for each location, we selected two different sets of users, corresponding to the upper-quartile and the lower-quartile, respectively, based on the users' activity in terms of total ratings on items belonging to the location. Note that no distinction was made between positive (high) and negative (low) ratings of the users, which clearly affected the overall quality of the average performance results: these were indeed generally low due to a partial agreement with the TripAdvisor-driven reference ranking generated for each location (cf. Sect. V-A), which by definition tends to rank higher positively rated items, and hence packages.

Looking at the results per location in each group, lower scores corresponded to average Fagin and MAP, which how-

TABLE IV
BEST PERFORMANCES (FAGIN AND MAP SCORES) OF PACKREC AND COMPOSITEREC OVER DIFFERENT LOCATIONS.

Location	PackRec		CompositeRec		Location	PackRec		CompositeRec	
	Fagin	MAP	Fagin	MAP		Fagin	MAP	Fagin	MAP
Amsterdam	.530	.659	.089	.090	PlayadelCarmen	.219	.254	.022	.028
Barcelona	.589	.841	.184	.307	Rome	.487	.735	.099	.089
BuenosAires	.528	.828	.112	.111	SanFrancisco	.486	.314	.193	.184
Chicago	.496	.839	.135	.124	SharmElSheikh	.466	.793	.092	.085
Edinburgh	.514	.586	.086	.069	Sydney	.493	.913	.119	.106
Honolulu	.520	.682	.072	.061	Venice	.607	.931	.229	.524
Istanbul	.589	.887	.101	.112	WashingtonDC	.296	.494	.057	.032
NiagaraFalls	.623	.954	.071	.072	*avg*	.496	.714	.111	.133

ever behaved quite closely. Generally higher scores were achieved in terms of Kendall, which would indicate a higher overall alignment w.r.t. the reference ranking than in the cases where the head of lists was taken into account, and in terms of nDCG, which means that the shared packages between the top-k list of PackRec and the top-k list of the reference ranking obtained similar ranks and relevance scores. By comparing the corresponding group average performances in the two cases of user activity, improved results could also be obtained in the lower-quartile case (i.e., for groups 3 and 4), where the sparsity in the user's ratings is generally higher.

PackRec also appeared to be moderately robust w.r.t. the various queries (i.e., package models) and location group settings, since the differences among the group averages were relatively low in terms of average Fagin and MAP. By analyzing the personalization of the recommendations obtained for different budgets (results not shown), we observed that in several cases smaller budgets corresponded to better top-rankings (i.e., higher Fagin scores, nDCG and MAP), which might be explained by the fact that smaller budgets led to filter out a larger number of packages, facilitating the ranking task.

Comparison with CompositeRec: Table IV shows the results obtained by PackRec and CompositeRec. Comparison was limited to the Fagin and MAP criteria for two main reasons: both the PackRec and the CompositeRec outputs are incomplete rankings w.r.t. the merged reference ranking described above (which prevents the use of Kendall), and the CompositeRec output ranking is produced without scores (which prevents the use of nDCG). Parameter k for Fagin was set to 10, since we let CompositeRec to recommend the top-10 packages for each of the selected users per location.

As we expected, CompositeRec produced packages significantly different from those involved in our first stage of evaluation: over the various locations, the number of distinct models varied from 5 to 10, and the size (i.e., number of item types) from 2 to 35 (average of 13.9). Moreover, packages with multiple items of the same type were also taken into account—while the average multiplicity was always close to 1, in each location there was however at least one item type with more than 10 occurrences in a package model.

Results in Table IV show that PackRec achieved significant correlation with the combined reference ranking in all locations. It came to our surprise that, despite PackRec was trained under a setting driven by CompositeRec recommendations, PackRec clearly outperformed CompositeRec in all cases and for both criteria, with average gains of 0.385 Fagin and 0.581 MAP. Qualitatively, worse performance by CompositeRec was indeed explained by the fact that most of the 10 packages returned by CompositeRec per location have poor match with the top-10 packages in the reference ranking. This would give evidence that PackRec can effectively deal with packages of varying structure and that its combination of expert finding, collaborative filtering and graph-based ranking allows for better performance than a cost/value knapsack strategy.

VI. CONCLUSION

We presented an application-independent framework for the recommendation of packages of items. While being fully unsupervised, the framework relies on a-priori (user-provided) knowledge on the structure and types of the packages to be learned and recommended in a personalized fashion. Testing on datasets of other domains is certainly needed to fully support our claim of application-domain independence and versatility. Moreover, besides deepening the evaluation of the PackRec sensitivity to the various parameters, we have identified a number of points to be studied, including the definition of a regularization framework to address a possible item correlation issue in the package learning step, and alternative graph models for the package ranking step.

REFERENCES

[1] T. Deng, W. Fan, and F. Geerts, "On the complexity of package recommendation problems," in *Proc. ACM PODS*, 2012, pp. 261–272.

[2] Q. Liu, E. Chen, H. Xiong, Y. Ge, Z. Li, and X. Wu, "A Cocktail Approach for Travel Package Recommendation," *IEEE TKDE*, PrePrints, doi.ieeecomputersociety.org/10.1109/TKDE.2012.233, 2012.

[3] R. Baraglia, C. Frattari, C. I. Muntean, F. M. Nardini, and F. Silvestri, "A Trajectory-Based Recommender System for Tourism," in *Proc. Conf on Active Media Technology (AMT)*, 2012, pp. 196–205.

[4] M. D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu, "Automatic construction of travel itineraries using social breadcrumbs," in *Proc. ACM HT*, 2010, pp. 35–44.

[5] M. Xie, L. V. S. Lakshmanan, and P. T. Wood, "Composite recommendations: from items to packages," *Frontiers of Computer Science*, vol. 6, no. 3, pp. 264–277, 2012.

[6] A. Angel, S. Chaudhuri, G. Das, and N. Koudas, "Ranking objects based on relationships and fixed associations," in *Proc. EDBT*, 2009, pp. 910–921.

[7] G. Koutrika, B. Bercovitz, and H. Garcia-Molina, "FlexRecs: expressing and combining flexible recommendations," in *Proc. ACM SIGMOD*, 2009, pp. 745–758.

[8] S. Lee, S. Song, M. Kahng, D. Lee, and S. Lee, "Random walk based entity ranking on graph for multidimensional recommendation," in *Proc. ACM RecSys*, 2011, pp. 93–100.

[9] S. E. Helou, C. Salzmann, S. Sire, and D. Gillet, "The 3A contextual ranking system: simultaneously recommending actors, assets, and group activities," in *Proc. ACM RecSys*, 2009, pp. 373–376.

[10] N. N. Liu and Q. Yang, "EigenRank: a ranking-oriented approach to collaborative filtering," in *Proc. ACM SIGIR*, 2008, pp. 83–90.

[11] M. Gori and A. Pucci, "ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines," in *Proc. IJCAI*, 2007, pp. 2766–2771.

[12] A. Brodsky, S. M. Henshaw, and J. Whittle, "CARD: a decision-guidance framework and application for recommending composite alternatives," in *Proc. ACM RecSys*, 2008, pp. 171–178.

[13] Q. T. Tran, C. Y. Chan, and G. Wang, "Evaluation of set-based queries with aggregation constraints," in *Proc. ACM CIKM*, 2011, pp. 1495–1504.

[14] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM TOIS*, vol. 20, no. 4, pp. 422–446, 2002.

[15] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing Top k Lists," *SIAM Journal on Discrete Mathematics*, vol. 17, no. 1, pp. 134–160, 2003.