Information Systems 48 (2015) 64–88

Contents lists available at ScienceDirect

Information Systems

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

# On recommendation problems beyond points of interest

Ting Deng [a,c], Wenfei Fan [b,c], Floris Geerts [d,*]

[a] School of Computer Science and Engineering, Beihang University Beijing, No. 37 XueYuan Road, 100191 Beijing, China
[b] School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, United Kingdom
[c] RCBD and SKLSDE Lab, Beihang University Beijing, No. 37 XueYuan Road, 100191 Beijing, China
[d] Department of Mathematics and Computer Science, University of Antwerp, Middelheimlaan 1, B-2020 Antwerpen, Belgium

ARTICLE INFO

*Article history:*
Received 27 February 2013
Received in revised form
27 August 2014
Accepted 31 August 2014
Recommended by: D. Suciu
Available online 16 September 2014

*Keywords:*
Recommendation problems
Query relaxation
Adjustment
Complexity

ABSTRACT

Recommendation systems aim to recommend items or packages of items that are likely to be of interest to users. Previous work on recommendation systems has mostly focused on recommending points of interest (POI), to identify and suggest top-$k$ items or packages that meet selection criteria and satisfy compatibility constraints on items in a package, where the (packages of) items are ranked by their usefulness to the users. As opposed to prior work, this paper investigates two issues beyond POI recommendation that are also important to recommendation systems. When there exist no sufficiently many POI that can be recommended, we propose (1) *query relaxation recommendation* to help users revise their selection criteria, or (2) *adjustment recommendation* to guide recommendation systems to modify their item collections, such that the users' requirements can be satisfied.

We study two related problems, to decide (1) whether the query expressing the selection criteria can be relaxed to a limited extent, and (2) whether we can update a bounded number of items, such that the users can get desired recommendations. We establish the upper and lower bounds of these problems, *all matching*, for both combined and data complexity, when selection criteria and compatibility constraints are expressed in a variety of query languages, for both item recommendation and package recommendation. To understand where the complexity comes from, we also study the impact of variable sizes of packages, compatibility constraints and selection criteria on the analyses of these problems. Our results indicate that in most cases the complexity bounds of query relaxation and adjustment recommendation are comparable to their counterparts of the basic recommendation problem for testing whether a given set of (resp. packages of) items makes top-$k$ items (resp. packages). In other words, extending recommendation systems with the query relaxation and adjustment recommendation functionalities typically does not incur extra overhead.

© 2014 Elsevier Ltd. All rights reserved.

## 1. Introduction

Recommendation systems are also known as recommender systems, recommendation engines and platforms. Such systems are widely used to identify and suggest information items (e.g., movies, TV, news, books) or social elements (e.g., people, friends, groups or events in social networks) that are likely to be of interest to users. Traditional recommendation systems aim to find top-$k$ items from a collection of items, e.g., books, events, Web sites and research papers [1], which satisfy certain selection criteria identified for a user, and are ranked by their

* Corresponding author.
*E-mail addresses:* dengting@act.buaa.edu.cn (T. Deng), wenfei@inf.ed.ac.uk (W. Fan), floris.geerts@uantwerpen.be (F. Geerts).

http://dx.doi.org/10.1016/j.is.2014.08.002
0306-4379/© 2014 Elsevier Ltd. All rights reserved.

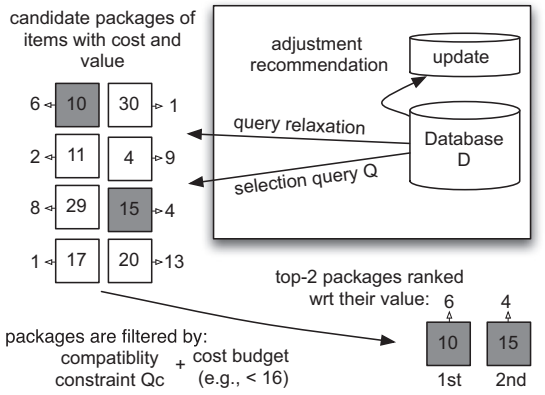


**Fig. 1.** Overview of package recommendation framework: Candidate packages get selected by query $Q$ from database $D$; each package comes equipped with a cost (shown inside the package) and value; valid (gray shaded) packages are selected based on compatibility constraints $Q_c$ and cost budget; and are finally ranked according to their value. In this paper, we add query relaxation and adjustment recommendation to the framework.

value of a utility function. More recently recommendation systems are often used to find top-$k$ packages, i.e., *sets of items*, such as travel plans [2], teams of players [3] and various course combinations and prerequisites [4–6]. The items in a package are required not only to meet the selection criteria for each *individual* item, but also to satisfy compatibility constraints defined on all the items in a package *taken together*. Packages may have variable sizes subject to a cost budget, and are ranked by overall ratings of their items determined by a utility function [2]. Relational queries are often used to specify selection criteria and compatibility constraints [6–8,4,2], as illustrated below. An overview of the recommendation framework considered in this paper is depicted and illustrated in Fig. 1.

**Example 1.** Consider a recommendation system for travel plans, which maintains two relations specified by

flight(f#, From, To, DT, DD, AT, AD, Pr),

vista(name, city, type, ticket, time, dates).

Here a flight tuple specifies flight f# from From to To that departs at time DT on date DD and arrives at time AT on date AD, with airfare Pr. A vista tuple specifies a site name to visit in the city, its ticket price, type (e.g., museum, theater), the amount of time needed for the visit; there is an entry for each range of dates for which it is open to the public.

(1) Item recommendation. One wants to find top-3 flights from EDI (Edinburgh) to NYC (New York City) with at most one stop, departing on 1/1/2013, with lowest possible airfare and duration time. This can be stated as an item recommendation problem: (a) flights are items; (b) the selection criteria are expressed as a union $Q_1 \cup Q_2$ of conjunctive queries (CQ), where $Q_1$ and $Q_2$ select direct and one-stop flights from EDI to NYC on 1/1/2013, respectively; and (c) the items selected are ranked by a utility function $f()$: given an item $s$, $f(s)$ is a real number computed from the airfare Pr and the duration Dur of $s$ such that the higher the Pr and Dur are, the lower the rating of $s$ is, where Dur can be derived from DT, DD, AT and AD, and $f()$ may associate different weights with Pr and Dur.

(2) Package recommendation. A user is planing a 5-day holiday, by taking a direct flight from EDI to NYC departing on 1/1/2013 and visiting as many places in NYC as possible. In addition, she does not want to have more than 2 museums in a package, which is a compatibility constraint [2]. Moreover, she wants the plans to have the lowest overall price.

This is an example of package recommendations: (a) the selection criteria are expressed as the following conjunctive query $Q$, which finds pairs of flight and vista tuples as items. That is, $Q$(f#, Pr, name, type, ticket, time, dates) is given by

$$\exists\, \mathsf{DT}, \mathsf{AT}, \mathsf{AD}, x_{\mathsf{To}} \big(\mathsf{flight}(\mathsf{f\#}, \mathsf{EDI}, x_{\mathsf{To}}, \mathsf{DT}, 1/1/2013, \mathsf{AT}, \mathsf{AD}, \mathsf{Pr}) \wedge \mathsf{vista}(\mathsf{name}, x_{\mathsf{To}}, \mathsf{type}, \mathsf{ticket}, \mathsf{time}, \mathsf{dates}) \wedge x_{\mathsf{To}} = \mathsf{NYC}\big);$$

(b) a package $N$ consists of some items that have the same f# (and hence Pr); (c) the rating of $N$, denoted by val($N$), is a real number such that the higher the sum of the Pr and ticket prices of the items in $N$ is, the lower val($N$) is; (d) the compatibility constraint requires that a package has no more than 2 museums, and can be expressed as another conjunctive query $Q_c$ such that $Q_c(N) = \emptyset$, where $Q_c$ is given by

$$Q_c() = \exists\, \mathsf{f\#}, \mathsf{Pr}, n_1, p_1, t_1, d_1, n_2, p_2, t_2, d_2, n_3, p_3, t_3, d_3 \big(R_Q(\mathsf{f\#}, \mathsf{Pr}, n_1, \mathit{museum}, p_1, t_1, d_1) \wedge R_Q(\mathsf{f\#}, \mathsf{Pr}, n_2, \mathit{museum}, p_2, t_2, d_2) \wedge R_Q(\mathsf{f\#}, \mathsf{Pr}, n_3, \mathit{museum}, p_3, t_3, d_3) \wedge (n_1 \neq n_2) \wedge (n_1 \neq n_3) \wedge (n_2 \neq n_3)\big).$$

Here $R_Q$ denotes the schema of the query answer $Q(D)$; and (e) the cost of $N$, denoted by cost($N$), is the total time taken for visiting all vista sites in $N$, which cannot exceed the time allocated for sightseeing in 5 days. Furthermore, cost($N$) assigns $+\infty$ whenever the package contains vistas that are not open during the 5 day holiday, by using the dates information; such packages will thus not be recommended. Putting these together, the travel planning recommendation system is to find top-$k$ such packages ranked by val($N$), for a constant $k$ chosen by the user. □

The need for studying recommendation systems is evident in Web services [2], Web search [9], social networks [9], education software [6] and commerce services [1]. There has been a host of work on recommendation problems, mostly focusing on algorithms for selecting and suggesting items or packages, known as *points of interest* (*POI*) [2], which meet selection criteria and satisfy compatibility constraints (see [1,9] for surveys). There has also been work on the complexity of computing POI recommendations [10,11,3,5,6,2].

There are other central issues beyond POI recommendation in connection with recommendation systems. In practice one often gets *no sensible recommendations*, i.e., the system fails to find items or packages that satisfy the user's needs. This may happen either when the selection criteria given by the user are too restrictive or when the

Download English Version:

https://daneshyari.com/en/article/396491

Download Persian Version:

https://daneshyari.com/article/396491

Daneshyari.com