Recommending Improved Configurations for Complex Objects with an Application in Travel Planning

Amihai Savir, Ronen I. Brafman
Department of Computer Science
Ben-Gurion University
{saviram,brafman}@cs.bgu.ac.il

Guy Shani
Information Systems Engineering
Ben-Gurion University
shanigu@bgu.ac.il

ABSTRACT

Users often configure complex objects with many possible internal choices. Recommendation engines that automatically configure such objects given user preferences and constraints, may provide much value in such cases. These applications generate appropriate recommendations based on user preferences. It is likely, though, that the user will not be able to fully express her preferences and constraints, requiring a phase of manual tuning of the recommended configuration. We suggest that following this manual revision, additional constraints and preferences can be *automatically* collected, and the recommended configuration can be *automatically* improved. Specifically, we suggest a recommender component that takes as input an initial manual configuration of a complex object, deduces certain user preferences and constraints from this configuration, and constructs an alternative configuration. We show an appealing application for our method in complex trip planning, and demonstrate its usability in a user study.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Recommender Systems, Reconfiguration, Trip planning

1. INTRODUCTION

The configuration of complex objects is known to be a difficult task [1]. Consider the task of constructing a trip to a country, where the goal of such a trip is to visit a set of attractions located in different parts of the country. A configuration of such a trip consists of a set of attractions together with a schedule for visiting them, as well as additional locations for over-night stays and meals.

To find a good travel plan, a traveler may research the destination country thoroughly, reading travel books or talking to past travelers. Although some people enjoy this planning phase, it is clear that constructing a satisfactory trip requires investing significant time and effort. Given limited time, it is impossible to exhaustively search the huge configuration space and design an optimal trip.

Alternatively, a soon-to-be traveler may use a sophisticated recommender system (e.g., [9]) for constructing trips. In these systems the user often answers a set of required questions about her preferences, and receives a recommended plan. This plan is unlikely to be taken as is by the user, and users typically revise the recommended plan manually, by, e.g., switching attractions, changing the schedule, and so forth. This manual revision phase is needed because it is well known that expressing preferences and constraints is difficult and exhausting for users [7]. Thus, one can consider the manual revision phase as implicitly expressing additional preferences and constraints that were not initially inserted.

We propose here a method that can be used to automatically improve such manually constructed or revised plans. We envision our component within a sophisticated travel recommender system, following the manual revision phase.

Our suggested algorithm takes as input a manual configuration from the user, with additional personal preference data. In the trip example, we the user provides an initial planned trip, possibly generated through a manual revision of a previously suggested trip. We also expect numeric ratings for attractions that appear within the trip, reflecting how interesting each chosen attraction seems to the user.

The system then learns from this input the user's preferences over both attraction properties and global trip properties. For example, from a user who rates many museums positively the system may learn that the user likes museums. The system also learns constraints over the relationships between items in the configuration. For example, the system may learn that the user plans a trip with no more than an hour drive between attractions, and hence deduce a constraint of short distances between attractions. The system automatically deduces preferences and constraints about both component classes and the relationship between components. Our algorithm then uses integer programming (IP) to create a better plan, which is returned to the user for inspection. The user may choose to farther revise the trip, and so forth.

We report a user study for trip planning to New Zealand. Participants in the user study, all people who toured New Zealand, provided an initial 5-6 day trip. The study participants clearly preferred the revised trip, and indicated that if a system like that was available, they would have liked to use it.

RecSys'13, October 12–16, 2013, Hong Kong, China.

2. BACKGROUND

Recommender systems [8] are now widely used in many applications. Perhaps the most popular approach for providing recommendations is the collaborative filtering (CF) approach requiring historical usage data. A popular alternative is the content-based (CB) approach [6], where items are described using a set of attributes. For example, trip attractions may be described by their category (e.g. museum, or hiking), by their location, by their price, and so forth.

In some cases, in addition to the information about item attribute values, the system must also maintain information about relationships between items. For example, in the case of attractions we may need to have distance information or public transportation opportunities between attractions. Two attractions in New Zealand, for instance, may be relatively nearby geographically, but due to a mountain range in between may require significant time to travel in between. Also, when visiting an attraction one must plan accommodations, dining, and so forth. The museum, the restaurant, and the hotel must be sufficiently close in order for the user to enjoy them all in one afternoon. Other information may describe how similar two attractions are. For example, two museums may be very similar, making a visit to both less desirable, or they could be distinct, and a visit to both can be interesting for a museum lover.

A system that maintains, in addition to the item attributes, relationships between attributes and items, is often referred to as a knowledge-based recommender system [5]. Such systems are naturally costly to build, as the knowledge must be identified by a domain expert, then collected and inserted into the system. Therefore, they are appropriate mainly to systems that sell relatively profitable items. Indeed, vacation packages are an obvious example of such complex and costly products that require much knowledge to construct properly, and it is no surprise that much work on knowledge-based recommenders considers this domain [9].

A second, orthogonal, research direction considers the interactions of the users with the system [9]. When recommending costly and complex items, it is unlikely that the system will provide good recommendations without collecting preference information from the user by, e.g., filling out a lengthy questionnaire. An alternative to a tiresome questionnaire is to present the user with recommended items after a very brief collection of user goals (e.g. the vacation destination). These recommendations are likely not to be satisfactory, and we must allow the user to criticize them. From the user's response, the system learns additional preference information, and provides improved recommendations.

Such systems provide relatively lengthy interactions between the user and the system to identify good items. Therefore, such systems are useful in the case where a user gains significant value from the items, and is hence willing to invest the required effort in a conversation with the system.

There are a few examples of previous systems designed to recommend travel destinations or plans to users (see, e.g. [4]), in addition to many online travel planning systems with no recommendation component, such as Travelocity or Trip Advisor. These systems typically ask the user to specify many input parameters, such as the destination, the travel party (e.g. couple or family), duration, budget, attraction types and so forth. Then, these systems may use information over the choices of other users to recommend attractions, accommodations and so forth.

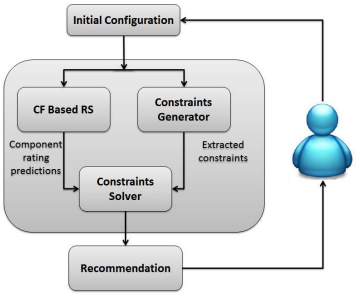

Figure 1: Complex Objects Recommender System.

3. TRAVEL PLANNING RECONFIGURATION

Figure 1 presents our complex objects recommender system, composed of 3 major components — a recommender, a constraint generator, and a constraint solver. The system takes as input an example of a possible configuration of a complex object designed by the user. It is important that the user will provide a sufficiently rich example, because the system then deduces from this example various features. Two separate types of features are learned, the user preference over components, and a set of constraints over the combination of components into a complex object. Then, the system uses the deduced features to compute a recommended complex object. This object is then returned to the user, and she may revise it manually, and provide it as input for a second recommendation.

Recommender: Takes as input a list of preferred or positively rated components and uses some rating prediction approach, such as CF, to predict ratings for other components. These ratings do not take into consideration any constraints or correlations between components.

Constraint generation: Identifies constraints expressed in the given example complex object. Some constraints, such as total price, are global, i.e., apply to the complete complex object. Other constraints apply to subsets of components. For example, the maximal time allotted in the trip to travel between attractions. Finally, there are some constraints that apply to specific components. For example, the user may decide that some attractions must be kept. In our system, all attractions that got the highest possible rating, are considered irreplaceable.

Constraint solver: We now feed the results of the two previous stages to an optimizer, designed to provide a solution that satisfies all constraints while maximizing the sum of predicted ratings. The solution directly corresponds to a complex object that meets all constraints and is considered best in terms of user preferences, e.g., the sum of ratings of all attractions, or some other aggregate measure.

While the system we suggest above is designed generically to fit various types of complex objects, in this paper we focus on a travel planning application. We consider a trip to be a complex object constructed of a sequence of attractions (the components of the complex object). The system's goal is to provide a trip, i.e., a schedule of activities for a 5 day visit to New Zealand. We now explain how the generic phases in the former section manifest themselves in our application.

Database: Our system requires a database consisting of the various possible components (attractions), their attributes, and their relationships. Our database contained 327 attractions, 21 attraction types, and 50 different areas

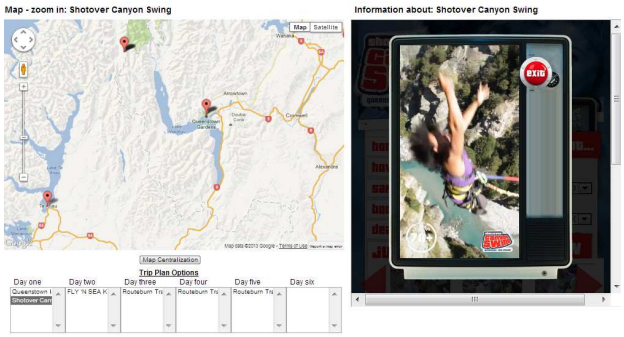

Figure 2: Trip suggestion user interface.

of New Zealand. All the information in the database was taken from the popular Lonely Planet travel guide [2].

Input: The interaction of users with the system begins with inserting information about a possible trip. In our application we asked the user to specify which attractions she would like to visit during each of the 5 days. In addition, the user specified an attraction interest rating, specifying how important is the attraction, on a scale of 1 to 5. Finally, the user was asked to provide information regarding the area where she would like to spend each night.

Attraction Recommender: In order to extrapolate attraction ratings for users, we used a simple CF engine. We constructed a memory-based algorithm, using the popular Pearson correlation to compare user profiles [3]. The algorithm was run on all the users who provided initial ratings for attractions in their input.

Trip constraint generation: We define the following constraints over the revised trip:

• Budget — we computed the overall price of the trip that the user has given us as input. We then require the budget of the revised trip not to exceed this budget by some fixed amount δ (in our experiment we used $\delta = 50NZD$).

• Daily load — the minimal and maximal time required for the attractions that the user chose to visit in a single day. We thus measure whether the user prefers a relaxed trip, or a packed trip, visiting as many attractions as possible.

• Travel distance — the maximal travel distance between attractions. This allows us to measure how much time the user is willing to spend traveling between places.

• Diversity level — input trips contain various types of attractions. We constrain the revised trip to also contain the same attraction types, maintaining the diversity of the trip. For example, even if the recommender predicts higher scores for all hiking attractions than all museum attractions, yet the user entered a few museum attractions in her input trip, we will force the revised trip to also contain some museums. We considered here only attractions rated 3 or higher.

• Preserving important attractions — in the input the user may specify a few attractions as "very important" (a rating of 5). We force the revised trip to contain these attractions.

• Local transportation constraints — while our system does not maintain transportation information between attractions, such as how to get from one museum to another within the same city, we add a fixed transportation time (30 minutes) for travel between attractions in the same city.

Trip constraint optimization: We formulate the problem of optimizing the total value of visited attractions subject to the above constraints using an integer programming (we used http://lpsolve.sourceforge.net/5.5/). To do so, we define two types of boolean variables — $x_{i,j}$ is true whenever attraction j is visited on day i, and $y_{i,l}$ is true whenever area l is visited in day i. We optimize $\sum_{i,j} R_j x_{i,j}$ where R_j is the predicted or specified rating for attraction j. That is, the program optimizes the sum of attraction ratings.

Any solution to the IP problem is mapped directly into a specification of which attractions to visit in each day, using the $x_{i,j}$ boolean parameters.

4. EMPIRICAL EVALUATION

We ran a user study with people who have visited New Zeland within the past 5 years. The users in our study were thus asked to specify a 5 day trip to New Zealand. They were asked to provide a sequence of attractions that they would like to visit in their trip, and to specify in which city they would spend each night. The users also had to add an interest rating for each attraction in the input trip.

These manually constructed trips were run through our system and we computed 3 different suggestions:

1. Best revision: the output of the algorithm of the previous section, within the given time limit.

2. Conservative revision: a revision with an additional set of constraints forcing at most two attractions that were ranked lowest to be replaced.

3. Best attraction list: a list of the attractions with the highest predicted rating, with no trip planning, as is done by simpler recommendation systems.

The alternatives were presented in a random order to reduce biasing due to presentation order. The users could observe the revisions, watch the suggested trip on the map, and request information from Wikipedia concerning the attractions in a separate pane on the user interface (Figure 2). The users were then asked to fill a short survey asking questions about the 3 suggestions.

We collect attraction ratings for our CF component thorough simple online application that allows travelers to New Zealand to provide ratings for attractions that they visited. This application was installed by the owners of a number of guest houses in New Zealand, which are popular among travelers. We collected more than 600 responses of travelers through this online application.

4.1 Experimental Results

We had 19 participants in our user study, ages 20 through 50. All of them have visited New Zealand in the past 5 years. We had 12 male participants and 7 females. Although this is certainly not a large user study, we still got significant results, which further suggests that the differences between the methods are very noticeable. The trips that were inserted were very diverse. Figure 3 shows the distribution of the 5 most popular attraction types. We can observe that the user population had museum lovers, and people who never visit museums, avid hikers alongside people who choose one or less hikes, and so forth. In addition, there were 7 attraction types, such as cruises, fishing, and horse riding, that were selected by only a single person.

As these extracted constraints demonstrate, a personalization approach is needed to understand the different choices that people make when constructing trips, and to be able to automatically construct trip revisions that will be appropriate for their personal taste.

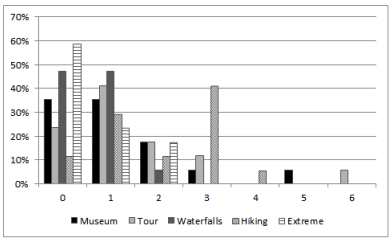

Figure 3: Popular attraction types distribution in input trips. The x-axis shows the number of attractions of that type per trip. The y-axis shows the portion of the user population who chose this number of attractions.

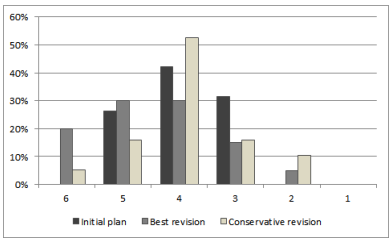

Figure 4: Grades distribution for the initial plan and the two revisions presented to the users.

These extracted preferences, along with the attraction ratings that the users supplied, were inserted into the system and the 3 suggestions were computed and presented to the users. After presenting the users with the suggestions, they were asked to fill a short survey.

We asked several general questions regarding the participants' perception of our system: 90% (17) of the participants said that trip planning is difficult and requires much effort, and 95% (18) of them thought that our recommender system for trip planning will be successful, if made available widely. 95% of the users claimed that given the system suggestions they would have made some change to their initial plan.

We now turn to an evaluation of the travel plans and their revisions. Figure 4 shows distribution of grades for the revisions. The average grade for the initial plan was 3.94, while the average for the best revision was 4.45, and the average for the conservative revision was 3.89. In 10 times out of 19, the best revision got a higher grade than the initial plan, and for 6 users both plans got the same grade. Only 3 users gave a higher grade to their initial plan. A sign-test gives a p-value of 0.002 to the null hypothesis that the revised plan is no better than the initial plan.

Only 6 participants marked the conservative revision to be better than the original plan, and 4 gave the original plan higher grades. We allowed the participants to comment on the plans, and we got responses criticizing the conservative plan as being not novel enough.

We also asked the participants which suggestion they preferred. 79% (15) preferred the best revision suggestion over the other two suggestions. 90% preferred a suggestion for a complete plan over a list of interesting attractions. These results demonstrate the value of supplying the actual target object, rather than useful components.

The construction of an initial plan represents a significant effort, and we asked whether the effort was worthwhile. Only one participant claimed that the effort was too high for the given results. One other participant said that the effort is high, but the results are interesting. 26% (5) of the participants said that the effort is considerable, but it is acceptable for trip planning, and the rest 63% (12) of the participants said that the effort is minor given the results.

Although all of our participants already visited New Zealand, 79% of them claimed that the system provided some novel suggestions of places they were unfamiliar with and looked interesting. Some people commented that they recognized some of the suggested attractions and considered them to be good suggestions, indicating the power of our recommender component.

5. CONCLUSION

We present a recommendation component that takes as input a pre-configured complex object, such as a trip to a country, and some preference data, such as attraction ratings, and produces a better configuration, e.g. an improved trip, taking into account a set of constraints that were extracted from the input example. In a user study over a 5 day trip planning to New Zealand, we produced better trips than manually constructed input trips.

In the future we plan to test our component within a complete recommender system, as an integral part of the trip planning process. In addition, we intend to test our ideas in other domains where complex objects need to be configured, such as configuring an appropriate computer system.

Acknowledgment: This work is partially supported by the Lynn and William Frankel Center for Computer Science.

6. REFERENCES

[1] M. Aldanondo, G. Moynard, and K. Hamou. General configurator requirements and modeling elements. In *ECAI Workshop on Configuration*, pages 1–6, 2000.

[2] C. Bain, G. Dunford, K. Miller, S. O'Brien, and C. Rawlings-Way. *New Zealand — Lonely Planet.* Lonely Planet, 2006.

[3] J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.

[4] T. Cao, Q. Nguyen, A. Nguyen, and T. Le. Integrating open data and generating travel itinerary in semantic-aware tourist information system. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, pages 214–221. ACM, 2011.

[5] P. Cunningham and A. Bonzano. Knowledge engineering issues in developing a case-based reasoning application. *Knowl.-Based Syst.*, 12(7):371–379, 1999.

[6] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. 2011.

[7] P. Pu, B. Faltings, L. Chen, J. Zhang, and P. Viappiani. Usability guidelines for product recommenders based on example critiquing research. In *Recommender Systems Handbook*, pages 511–545. 2011.

[8] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In *Recommender Systems Handbook*, pages 1–35. 2011.

[9] A. Venturini and F. Ricci. Aplying trip@dvice recommendation technology to www.visiteurope.com. In *ECAI*, pages 607–611, 2006.