Mining User Trails in Critiquing Based Recommenders

Skanda Raj Vasudevan
Department of Computer Science and Engineering
IIT Madras
Chennai-600036, India
skandavs@cse.iitm.ac.in

Sutanu Chakraborti
Department of Computer Science and Engineering
IIT Madras
Chennai-600036, India
sutanuc@cse.iitm.ac.in

ABSTRACT

Critiquing based recommenders are very commonly used to help users navigate through the product space to find the required product by tweaking/critiquing one or more features. By critiquing a product, the user gives an informative feedback(i.e, which feature needs to be modified) about why they rejected a product and preferred the other one. As a user interacts with such a system, trails are left behind. We propose ways of leveraging these trails to induce preference models of items which can be used to estimate the relative utilities of products which can be used in ranking the recommendations presented to the user. The idea is to effectively complement knowledge of explicit user interactions in traditional social recommenders with knowledge implicitly obtained from trails.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Filtering

Keywords

Recommenders; Critiquing; PageRank; Similarity; Utility

1. INTRODUCTION

Social recommenders benefit from the knowledge of associations between users and explicit feedback users provide on items they consume. The feedback can come in the form of ratings,votes,tags or textual comments. However, a significant component of knowledge about user relatedness and feedback is implicit in the data capturing interaction of users with the system. In most conversational recommender systems, users leave behind trails as they explore the product space. The essential contribution of our work is to demonstrate the potential of knowledge mined from such trails in terms of reducing the number of interaction cycles required to reach the product of interest.

WWW'14 Companion, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.
http://dx.doi.org/10.1145/2567948.2579238.

We focus on critiquing based recommenders which are commonly used to guide the user through the product space by facilitating interaction in the form of critiques. Here the user views a product and may give a critique like *show a similar product but a cheaper one*, to which the system responds by recommending the products that satisfy the critique and are most similar to the current product. The motivation behind critiquing in recommender systems is that it is easier for users to critique a product than construct formal queries.

In the context of critiquing based recommenders, a trail can be defined as the path the user followed starting from a product, critiquing and choosing the preferred product in each cycle until he/she reaches the product he/she is satisfied with, which is generally termed as the target. In a restaurant domain where $r_i's$ are restaurants, an example trail is:

$$r_1 \xrightarrow{cheaper} r_3 \xrightarrow{nicer} r_6 \xrightarrow{creative} r_2 \xrightarrow{cheaper} r_5$$

The holy grail in designing recommender systems is to arrive at a robust estimate of the utility of a product with respect to a user and a set of expressed preferences. While similarity is often used as a surrogate for utility, it is at best an a priori approximation closely tied to choices made in representation and similarity measures. When users critique a product and make choices, they implicitly assess utilities. Hence, critiques from sizeable logs of trails can be used to induce a model of relative utilities, which can be used to rank recommendations and enhance user experience. We show the effectiveness of a class of PageRank-style algorithms in this context.

For our experiments we considered *Entree*, a restaurant recommendation system [1, 3]. Here the critiques are defined over high level features like *nicer, livelier, creative* etc. Users browse through the restaurants by redirecting the search with the help of critiques like the ones mentioned above until he finds a restaurant that he is satisfied with.

2. RELATED WORK

FindMe systems developed by Burke & co.[3] are the earliest critiquing based recommenders. From then on critiquing based recommenders have been studied from various perspectives like ease of interaction, different critiquing techniques to improve the effectiveness of search and reducing the user effort to reach a target[5]. Some of the earlier work in these types of recommenders considered analyzing user logs to enhance recommendations. In [2], a hybrid recommender is proposed adding collaborative flavor to the critiquing. Here user trails in critiquing based recommenders

are used to construct the user-item rating matrix which is essential in a collaborative setting to give recommendations. In [8], a case based reasoning setting is adapted to analyze the user sessions and reuse the past critiquing experiences to improve recommendations. Here each session that is successful in the past (i.e, where user reached a target) is converted into a problem-solution pair and this forms a case. The problem part of the case is the path user followed to reach the target while the solution is the target product of the session. For a given new session the similar sessions are identified and corresponding solutions are ranked and presented to the user. Here the similarities between sessions are estimated based on the overlap of critiquing patterns between them. However, here estimating similarities is based only on the critique patterns and types of items that the user has been recommended so far are not considered due to which sometimes unexpected recommendations are shown. To overcome this, in [10] a new component is added which takes into account item similarity when selecting relevant sessions. Modeling user preferences using graphs and estimating utilities was studied in [6] in the context of preference based recommenders. Here the domain specific dominance knowledge is combined with SimRank based similarity to give recommendations. The utilities are estimated using PageRank on the dominance graphs formed during each cycle of user interaction. Besides recommender systems, trails are extensively studied in the context of web search and retrieval. In [4, 11] clickthrough logs of users are studied to understand user behavior and improve search performance and also help users in navigating through the web.

3. OUR APPROACH

3.1 Entree: A brief overview

Entree is a restaurant recommendation system for the city of Chicago[3, 2]. A user interacts with the system by submitting an entry point, either a known restaurant or a set of criteria and the system responds with "similar" restaurants. In [3] a sketch of how similarities are estimated is given. By interactively redirecting the search using the critiques a user finds an acceptable option. Critiques used in *Entree* are *Cheaper, Creative, Lively. Nicer, Quieter, Traditional.* Our experiments are based on the *Entree* dataset [1].It consists of trails, where each trail is a sequence of user interactions while reaching the target. In *Entree* once a critique is given the system responds with maximally similar restaurants satisfying the critique. The user can browse through the list and select one for the next critiquing cycle or stop the trail indicating he is satisfied with it. An example of such a trail is shown in Table 1. Initially user critiques r_1 with *cheaper*

Table 1: Structure of a trail in *Entree*

r_1	$\xrightarrow{cheaper}$	r_3	$\xrightarrow{browse}$	r_{10}	$\xrightarrow{browse}$	r_2	$\xrightarrow{cheaper}$
r_5	$\xrightarrow{cheaper}$	r_9	$\xrightarrow{nicer}$	r_7	$\xrightarrow{browse}$	r_8	$\xrightarrow{browse}$
r_{23}	$\xrightarrow{livelier}$	r_{12}					

and prefers r_2 after viewing r_3 and r_{10}. This sub-trail constitutes one critiquing cycle. The number of sub-trails en route r_1 through r_{12} is five; note that this is the path length discounting the edges labeled *browse*. After a few cycles of critiquing the session comes to an end with the user reaching a target or giving up.

3.2 Preference graphs

In the trail shown in Table 1, the user preferred restaurant r_2 as a *cheaper* alternative to restaurant r_1 among the given recommendations, similarly she rated r_{23} *nicer* than r_9 and r_4 *livelier* than r_{23}. We use preference graphs to represent these preferences. A preference graph is constructed for each critique by aggregating all trails. Examples of such graphs are shown in Fig 1 and these formed based on the four trails shown in Table 2. In the next section we describe ways of using these preference graphs in ranking the recommendations.

Table 2: Examples of trails indicating the preferences: here B-*browse*, C-*cheaper*,L-*livelier*

T_1	$r_1 \xrightarrow{C} r_3 \xrightarrow{B} r_{10} \xrightarrow{B} r_2 \xrightarrow{C} r_5 \xrightarrow{C} r_9 \xrightarrow{N} r_7 \xrightarrow{B} r_8 \xrightarrow{B} r_{23} \xrightarrow{N} r_{14} \xrightarrow{L} r_{12} \xrightarrow{B} r_4$
T_2	$r_5 \xrightarrow{C} r_3 \xrightarrow{B} r_8 \xrightarrow{B} r_6 \xrightarrow{B} r_9 \xrightarrow{N} r_{10} \xrightarrow{B} r_{23} \xrightarrow{N} r_{14} \xrightarrow{L} r_2 \xrightarrow{B} r_{15} \xrightarrow{B} r_4$
T_3	$r_2 \xrightarrow{C} r_3 \xrightarrow{B} r_5 \xrightarrow{B} r_9 \xrightarrow{B} r_6 \xrightarrow{B} r_{14} \xrightarrow{L} r_4 \xrightarrow{L} r_9 \xrightarrow{N} r_5 \xrightarrow{N} r_7 \xrightarrow{B} r_1 \xrightarrow{N} r_{23}$
T_4	$r_9 \xrightarrow{L} r_1 \xrightarrow{L} r_5 \xrightarrow{C} r_6 \xrightarrow{B} r_3 \xrightarrow{B} r_{14} \xrightarrow{C} r_1 \xrightarrow{C} r_2 \xrightarrow{N} r_4 \xrightarrow{N} r_1 \xrightarrow{N} r_5$

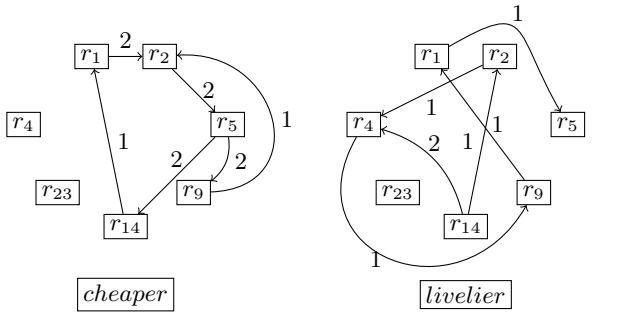


Figure 1: Preference graphs for *cheaper* and *livelier*, here the weight of each edge represents the number of times destination is preferred after critiquing the source (some isolated nodes are not shown here).

3.3 Ranking Algorithms

In this section we propose four methods for ranking *Entree* recommendations based on preference graphs. The trail log is partitioned into training and test data; the goal is to use preference graphs to induce a preference model over training data and use the same to rank recommendations for each test sub-trail (refer Section 3.1), once users provide critiques.

3.3.1 Frequency based ranking(FR)

This is a baseline approach where the recommendations are ranked based on the frequency with which they are preferred over the current restaurant when the same critique is applied over the latter in the trail corpus. Given the training data in Table 2, consider a test case where a restaurant r_{14} is shown and the user applies the critique *livelier*. If *Entree* recommends r_2, r_4 *and* r_9, then a frequency based ranking

induces an order $<r_4, r_2, r_9>$. This is because the graph for *livelier* (Fig. 1) suggests that r_4 is preferred two times over r_{14} and r_2 is preferred once, but r_9 is never preferred over r_{14}.

3.3.2 Critique specific PageRank(CPR)

Consider the simple preference sub-graph in Fig. 2, if r_{14}

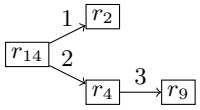


Figure 2

is the restaurant being critiqued in the test trail, we note that the FR algorithm gives r_4 as the best recommendation since it is preferred more frequently over r_{14} in the training data. Despite the fact that r_9 is never explicitly preferred over r_{14}, it seems intuitive to promote the ranking of r_9, since it is preferred thrice over r_4. This observation is of central to Critique specific PageRank (CPR), where the hypothesis is: *'The utility of a restaurant is high across a certain critique, if it is preferred over high utility restaurants over the same critique'*

This formulation has its close analogue in that of PageRank[9]. The PageRank algorithm assigns a real number to each node in a graph with an intent that the higher the PageRank score of a node, the more "important" it is. PageRank is generally used in web scenario to measure the importance of web pages based on the links leading to a page. With a small modification of the standard PageRank approach, it can be adapted for weighted graphs as well. The utility score is computed as shown below:

$$U(r_i) = \sum_j U(r_j) \times \frac{w_{ji}}{W_j} \quad (W_j = \sum_k w_{jk}) \qquad (1)$$

where $U(r_i)$ denotes utility of r_i, w_{ji} is the weight of edge from r_j to r_i. It is important here to note that these utilities are local to the critiquing dimension i.e, if we consider the graph for *nicer* and do a PageRank computation on it, the scores we get for each restaurant is the utility of that restaurant in the *nicer* dimension. Given a restaurant and its critique in a test sub-trail, CPR ranks *Entree* recommendations based on their PageRank utility scores across that critique. Note that unlike FR, CPR takes into account higher order associations between restaurants in the preference graph.

3.3.3 Restaurant and Critique specific PageRank (RCPR)

The recommendations generated by the CPR method are agnostic to the specific restaurant in the test trail that is critiqued. In this section we propose a method of making PageRank sensitive to each restaurant. Here preference graphs of critiques are formed specifically for each restaurant. There is an important difference in the way edges are added to the graph. Consider a critiquing cycle of a trail from r_1 through r_6,

$$r_1 \xrightarrow{cheaper} r_2 \xrightarrow{browse} r_4 \xrightarrow{browse} r_3 \xrightarrow{browse} r_6 \xrightarrow{livelier} ....$$

here for critique *cheaper* against r_1, r_6 is preferred over r_2, r_4, r_3 and hence we add the corresponding edges (r_2, r_6), (r_3, r_6), (r_4, r_6) to the graph of critique *cheaper* corresponding to r_1. In this manner the graphs are formed by aggregating preferences from all the trails. Some examples of restaurant and critique specific preference graphs are shown in Fig. 3 based on the trails in Table 2. Once these graphs

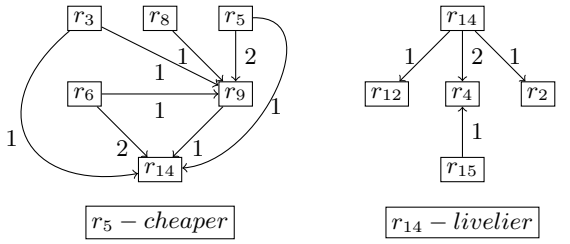


Figure 3: Preference graphs of r_5 for *cheaper* and r_{14} for *livelier*(some isolated nodes are not shown here).

are formed, the PageRank procedure is used to induce an ordering among the restaurants which can in turn be used in ranking the recommendations.

3.3.4 Fusion of CPR and RCPR

This method is a linear combination of CPR and RCPR.

$$Fusion_{score} = \alpha * RCPR_{score} + (1 - \alpha) * CPR_{score} \qquad (2)$$

where the value of α is determined using cross-validation. RCPR, unlike CPR, is sensitive to the critiqued restaurant. However, RCPR suffers from the problem of sparsity i.e, there may not be enough trails specific to a restaurant to induce ordering over a significant fraction of *Entree* recommendations. The fusion approach attempts to make the best of both worlds, by using CPR for smoothing the preference models and compensate for RCPR sparsity.

4. EVALUATION

From the *Entree* dataset used for our experiments we do not have access to the information about the complete list of recommended restaurants in each critiquing cycle. All we know from the trails are the restaurants that users browsed before picking one. One way of empirically testing the effectiveness of our approach would be to use the standard offline approach as in [7]. However, that would require us to have access to *Entree*'s similarity measures and not make best use of the actual user interactions that we have access to. We thus came up with an alternate evaluation scheme tailored to critique trails.

Evaluation criteria: In each sub-trail, we see if the order induced by a ranking algorithm helps in reducing the number of steps needed to reach the preferred restaurant. Supposing that the restaurants are ranked by the algorithm and presented to the user, the rank of the one finally picked by the user is indicative of the effectiveness of the approach. If the user viewed k restaurants before reaching the preferred one, the rank r of the preferred restaurant among these k restaurants as induced by the algorithm is observed. If the rank r is less than k then the number of steps that can be reduced is k - r.

Dataset: The dataset consists of user interaction sessions with the system. It has about 50672 trails collected over a period of 3 years. For all our experiments we split the dataset into train and test partitions. Utilities are estimated

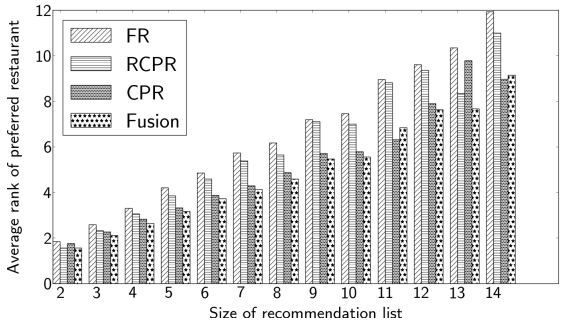
over the train data using approaches in Section 3 and their effectiveness over the test trails is evaluated. The results are reported after averaging across 10 different train and test partitions.

4.1 Results and Observations

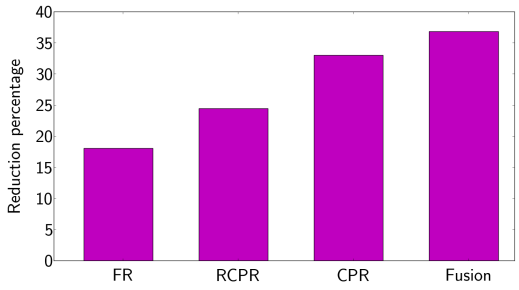
The performance of different methods described is shown in Fig. 4. In Fig. 4a, results are separately shown based on the number of recommendations user browsed before reaching the preferred restaurant. In Fig. 4b the overall reduction in number of steps by various methods is compared. The reduction in percentage of steps is computed by the formula,

$$\% \; reduction = \frac{k - r}{k} \times 100 \tag{3}$$

where k is size of recommendation list, r is number of steps needed to reach preferred restaurant. The results show that



(a) The results are divided into bins based on the number of recommendations(k) browsed before reaching target.



(b) Overall percentage reduction in number steps needed to reach preferred restaurant

Figure 4

RCPR is doing better than FR but not as good as CPR. This is due to the sparseness associated with the preferences available to each restaurant i.e, the restaurant and critique specific graphs are sparse. Also, it is important to note that though CPR does not explicitly take into account the restaurant being critiqued, this limitation is, to some extent, being compensated by the fact that in all our experiments ranking is only restricted to products reckoned by Entree as similar to the critiqued restaurant. Hence the Fusion of CPR and RCPR is performing the best. From our experiments the best value of α in Fusion method is observed to be 0.9.

5. CONCLUSION AND FUTURE WORK

Social recommenders that integrate explicit user feedback on items can benefit from integrating knowledge mined from users' implicit interactions with the system as well. This paper proposes and compares approaches designed with the goal of improving user experiences by mining relative critique specific utilities of items from user trails. We proposed a simple evaluation scheme centered around actual user behavior recorded in logs and empirical results demonstrate the effectiveness of the proposed approaches in reducing the number of interaction cycles. The work reported here is a stepping stone towards building plugins that can improve ranking produced by social recommenders by exploiting user trails. As a part of future work we would like to study this in a personalized setting and also add a collaborative flavor to the recommendation process [2]. The scope can also be extended by looking at trails outside those that are obtained from critiquing.

6. REFERENCES

[1] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[2] R. D. Burke. Hybrid recommender systems: Survey and experiments. *User Model. User-Adapt. Interact.*, 12(4):331–370, 2002.

[3] R. D. Burke, K. J. Hammond, and B. C. Young. The findme approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997.

[4] B. Cao, D. Shen, K. Wang, and Q. Yang. Clickthrough log analysis by collaborative ranking. In *AAAI*, 2010.

[5] L. Chen and P. Pu. Critiquing-based recommenders: survey and emerging trends. *User Model. User-Adapt. Interact.*, 22(1-2):125–150, 2012.

[6] S. Gupta and S. Chakraborti. Utilsim: Iteratively helping users discover their preferences. In *E-Commerce and Web Technologies*, pages 113–124. Springer, 2013.

[7] K. McCarthy, J. Reilly, L. McGinty, and B. Smyth. On the dynamic generation of compound critiques in conversational recommender systems. In *AH*, pages 176–184, 2004.

[8] K. McCarthy, Y. Salem, and B. Smyth. Experience-based critiquing: reusing critiquing experiences to improve conversational recommendation. In *Case-Based Reasoning. Research and Development*, pages 480–494. Springer, 2010.

[9] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report;, Stanford University, 1998.

[10] Y. Salem and J. Hong. History-aware critiquing-based conversational recommendation. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 63–64, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[11] A. Singla, R. White, and J. Huang. Studying trailfinding algorithms for enhanced web search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 443–450. ACM, 2010.