UtilSim: Iteratively Helping Users Discover Their Preferences

Saurabh Gupta and Sutanu Chakraborti

Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai, India - 600036
{sgupta,sutanuc}@cse.iitm.ac.in

Abstract. Conversational Recommender Systems belong to a class of knowledge based systems which simulate a customer's interaction with a shopkeeper with the help of repeated user feedback till the user settles on a product. One of the modes for getting user feedback is *Preference Based Feedback*, which is especially suited for novice users(having little domain knowledge), who find it easy to express preferences across products as a whole, rather than specific product features. Such kind of novice users might not be aware of the specific characteristics of the items that they may be interested in, hence, the shopkeeper/system should show them a set of products during each interaction, which can constructively stimulate their preferences, leading them to a desirable product in subsequent interactions. We propose a novel approach to conversational recommendation, UtilSim, where utilities corresponding to products get continually updated as a user iteratively interacts with the system, helping her discover her hidden preferences in the process. We show that UtilSim, which combines domain-specific "dominance" knowledge with SimRank based similarity, significantly outperforms the existing conversational approaches using *Preference Based Feedback* in terms of recommendation efficiency.

Keywords: Knowledge based Recommendation, Preference Based Feedback, Utility estimation, Case Based Recommendation.

1 Introduction

Imagine a prospective camera buyer who actually has very little domain knowledge about cameras. Due to lack of information about product characteristics, it becomes difficult for her to express her preferences adequately/fluently at the start of her interaction with the system[1]. She might not be aware of the product features/attributes that she may be interested in (due to lack of domain knowledge). Therefore, the recommender system should show her appropriate products spread across multiple interactions, which can help stimulate her preferences and lead her to an acceptable product. Typically, knowledge based recommendation systems estimate utility of a product with respect to a given query(or a reference product) by using a weighted linear combination of the local similarities(usually defined by experts) between the features of the query(or the reference product) and the product concerned[2]. We refer to this model as the *weighted similarity model* . Among the knowledge based approaches, *single shot retrieval* approaches assume that there is a fixed set of weights(importances) for each attribute and use these weights to compute the utility of each product for each type of user. *Single shot retrieval* is of limited use for novice or non expert users who find it difficult to express their preferences beforehand in the form of a query, as described earlier in this section. If the user is not satisfied with what she is served by the system, then she has to revise the initial query and start from scratch again. Also, even involving domain experts to get a handle on the weights used by these systems cannot be foolproof, because the weights are user specific. As an alternative to *single shot retrieval*, *conversational recommender systems* try to simulate the kind of interaction which usually happens between a user and a shopkeeper with the goal of minimizing the cognitive load experienced by the user. Cognitive load can be defined as the effort on the part of the user while interacting with the recommender system. Conversational systems provide a way to capture user feedback at varying levels of granularity which can iteratively help the recommendation system to get a handle on the hidden needs/preferences of the user. Different user feedback mechanisms based on the decreasing order of cognitive load experienced by the user as summarized by [3] are: *Asking questions directly from the user*[4], *Ratings based user feedback*[5], *Critique based feedback*(user puts constraints on features) [6,7,8,9,10] and *Preference based feedback*(user selects one product over the others).

Case Based Recommender systems generally use the *weighted similarity model* to estimate utilities of products. One drawback associated with range normalized local similarity measures commonly used in the *weighted similarity model* was pointed by [11] who argue that in view of the different types of range normalization(narrow or wide) used for computing local similarities, it is not correct to assume that the way these local similarity measures estimate similarity is equivalent. Another problem with defining local similarity measures is the knowledge engineering effort involved.

In this paper, we propose a new conversational recommendation approach, **UtilSim**, which dynamically updates product utilities while interacting with users. UtilSim integrates two kinds of knowledge to provide effective recommendations - a) domain specific “dominance” knowledge across attributes, which is user invariant and easy to acquire. For example - $Price = 20$ “dominates” (is better than) $Price = 40$. b) SimRank[12] based similarity, which keeps getting robust with more data and does not involve some of the drawbacks associated with traditional similarity measures as discussed earlier.

2 Related Work

Several knowledge based recommendation techniques rely on some notion of *weighted similarity* to calculate the utility of a product for a user. *Compromise driven retrieval(CDR)*[13], a single shot retrieval scheme, uses a notion of compromise to better reflect the user’s needs. Compromises may be defined as the ways in which the retrieved products differ from the user’s requirements(specified query). Since it does single shot retrieval, *CDR* would work well in scenarios when the user is well informed about the domain and has her preferences clearly defined in her head. In contrast, our algorithm is incremental and adaptive in nature and is able to help non expert users as well (who do not have their preferences defined clearly in their head) to reach an acceptable product.

More Like This(MLT)[14] is a commonly used strategy using *Preference Based Feedback*(user prefers one item over the others), with the user selecting a product P during every interaction cycle. In the next iteration, the user is shown those products that are most similar(according to *weighted similarity model*) to P. *False-leads* is a problem which plagues the MLT approach. Using all features of the selected product as the next query might be a bad idea if some of the features of the selected product were irrelevant for the user[15]. *wMLT* [14] dynamically weighs *attribute values* while interacting with the user, based on the difference of the *attribute value* of the selected product to those of the rejected products. UtilSim differs from *wMLT* in that it uses a SimRank based notion of similarity as opposed to the *weighted similarity model* . Also, it uses domain-specific "dominance" knowledge coupled with PageRank[16] to compute utility of a particular **attribute value**. *Adaptive Selection(MLT-AS)*[15], another conversational strategy based on *Preference Based Feedback* uses *weighted similarity model* along with a diversity component and ***preference carrying*** mechanism to effectively focus the recommender system in an appropriate region of the product space. On the other hand, in addition to not using *weighted similarity model*, UtilSim does not include an explicit diversity metric in its utility computation and the utility associated with *attribute values* is dynamic. While ItemRank[17] uses collaborative data to establish links between different movies, UtilSim uses "dominance" knowledge across attributes to establish links between different *attribute values*. The notion of "dominance" has earlier been used to define *Simple Dominance Model*[18] which tries to identify and explain such decision phenomena as Asymmetric Dominance Effect[19] and Compromise Effect[20]. Our approach, on the other hand, uses "dominance" knowledge to implicitly infer the individual feature-value utilities during user interaction.

3 Our Approach

Numerous feedback mechanisms have been used to capture user preferences in conversational recommender systems. It was shown in [21] that expert users were more satisfied with attribute based feedback elicitation methods whereas novice users considered it more useful to express their preferences on products as a whole, from which attribute preferences can then be implicitly computed. In scenarios where the users are not well versed with the domain, they might find it difficult to put constraints at the level of specific attributes as required by *critiquing*. With a view towards improving recommendation quality in these scenarios, our approach, UtilSim, uses a *Preference based feedback* strategy where the user just expresses a preference for one product over the others. After receiving feedback from the user, the system now has to construct a revised model of the user's preferences to account for the dynamic changes in user preferences.

SimRank Based Similarity: To compute product-product as well as *attribute-value* - *attribute-value* similarities, we construct a bipartite graph consisting of products and their particular *attribute values* as shown in Figure 1. If an *attribute value* is present in a particular product, then a link is created from that *attribute value* node to the product node. The nodes A, B and C in Figure 1 refer to three cameras in our database. The other nodes represent *attribute values* with M denoting $Memory$, R denoting $Resolution$

and P denoting $Price$. The numbers that follow M,R and P are the values of the respective attributes. For e.g.- R6 means that $Resolution$ equals 6. To infer similarities from the product data, we use the main idea in SimRank - two objects are similar if they are related to similar objects[12]. Hence, we can say that products are similar if they have similar attribute values and attribute values are similar if they are present in similar products. This kind of circularity leads to a recursive definition for similarity computation which is defined for bipartite graphs in [12] as:

$$sim(A,B)=\frac{C}{|I(A)||I(B)|}\sum_{i=1}^{|I(A)|}\sum_{j=1}^{|I(B)|}sim(I_i(A),I_j(B)) \tag{1}$$

$$sim(a,b)=\frac{C}{|O(a)||O(b)|}\sum_{i=1}^{|O(a)|}\sum_{j=1}^{|O(b)|}sim(O_i(a),O_j(b)) \tag{2}$$

where C is a constant and $I_i(A)$ represents the i^{th} in-neighbour of A and $I_j(B)$ represents the j^{th} in-neighbour of B. $O_i(A)$ represents the i^{th} out-neighbour of A and $O_j(B)$ represents the j^{th} out-neighbour of B. In Figure 1, $P300$ is an in-neighbour of A and A is the out-neighbour of $P300$. Figure 2 shows how the flow of similarity takes

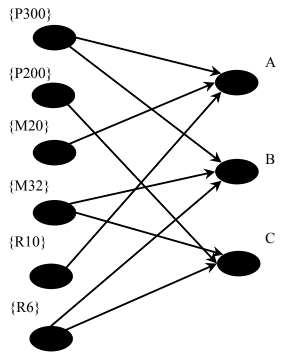


Fig. 1. Graph G having links from attribute values to products

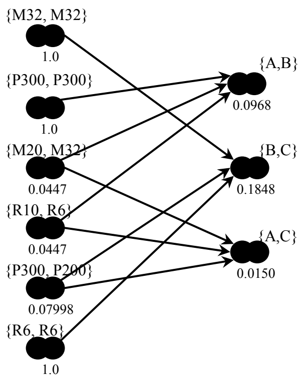


Fig. 2. Node pair graph G^2 with SimRank scores for C=0.8

place between the product-product nodes and the *attribute value* - *attribute value* nodes. This representation allows us to calculate similarities between *Price* = 200 and *Price* = 300 as well as product A and product B simultaneously.

Dominance Criteria across Attributes: We adopt the criteria used by [13] and divide the numerical attributes into two sets - *more is better* (MIB) and *less is better* (LIB). For the Camera dataset, we classify attributes *Price* and *Weight* as LIB attributes which implies that a lesser value of these attributes dominates a greater value. All the other attributes - *Optical Zoom*, *Digital Zoom*, *Resolution*, *Memory Included* are considered as

MIB attributes which implies that a greater value of these attributes dominates a lesser value. The idea of using MIB and LIB as the criteria for dominance can be motivated through the following example: A camera with *lower Price* and *higher Resolution* will always be preferred to a camera with *higher Price* and *lower Resolution*, all else being equal. Similarly, for the PC dataset, we classify *Processor Speed*, *Monitor*, *Memory*, *Capacity* as MIB attributes and *Price* as an LIB attribute. For example : $Price = 200$ dominates $Price = 500$. We make an assumption about the rationality of the user that if she selects product A, it must have at least one attribute a_i, which "dominates" the corresponding attribute b_i of at least one of the rejected products B. Note that in principle, not every product attribute would be monotonic. There might be attributes whose middle values are preferred to the extreme values. But, we think that those attributes can also be handled in a similar manner by transforming them through a function so that they reduce to being monotonic. For example, let us assume that an attribute X has values in the range 0 to 24, with the value of 12 being the most preferred. Therefore, we can transform X into

$$Y = 1 - \frac{|12 - X|}{12} \tag{3}$$

with Y being treated as an MIB attribute.

Dominance across nominal attributes is calculated as follows: Suppose, the user is currently at the R^{th} iteration in her interaction with the system. Considering a nominal attribute like $Manufacturer$ for a camera, let S denote the list of all the values corresponding to the $Manufacturer$ attribute for all the $R - 1$ selected products till the current iteration. In the list S, the latest(last) value corresponds to the $Manufacturer$ value of the product selected in the $(R - 1)^{th}$ iteration, in that order. For any two products a and b in the current recommendation set, we say that a_M dominates b_M, where a_M and b_M denote the $Manufacturer$ value of product a and b respectively if:

$$\Big[\Big(\sum_{k \in S} \alpha \cdot sim(a_M, k)\Big) - \Big(\sum_{k \in S} \beta \cdot sim(b_M, k)\Big)\Big] \geq 0 \tag{4}$$

where sim corresponds to the SimRank based similarity. The values of α and β keep on changing based on the position of k in S. If the position of k is towards the end of S, the values of α and β are high as compared to the scenario in which k is near the front of the list. These values are so kept to give higher weights to values which were selected more recently than to those which were selected during the initial part of the interaction.

3.1 UtilSim

We now explain how UtilSim computes revised product utilities after each user interaction through the following example. Assuming that products 1, 2 and 3 from Table 1 are the most similar(according to the weighted similarity model) products to the user query, the recommender system shows to the user these three products during the initial interaction. Notice that the use of weighted similarity model is a one time affair at the start of each dialogue after which it is never used in our approach. In principle, even for the first recommendation cycle we can generate recommendations using only SimRank

Table 1. Camera Models in a Shop

Product	Price	Resolution	Memory
1	300	10.0 MP	20
2	400	6.0 MP	32
3	200	8.0 MP	18
4	500	12.0 MP	32
5	600	12.1 MP	48

based similarity, but for all purposes in this paper, we used weighted similarity to start off the interaction process. Assuming that the user chooses product 3, we start off by giving high importance to all the *attribute values* of product 3 by creating links from *Price* value of product 1 to *Price* value of product 3 and from *Price* value of product 2 to *Price* value of product 3, repeating the same procedure for attributes *Resolution* and *Memory* as shown in Figure 3.

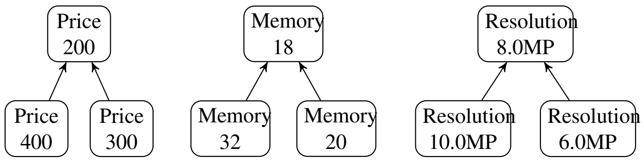


Fig. 3. Links formed due to the global dominance of selected product

But it may not be the case that the user liked all the *attribute values* of product 3 vis a vis the other products.

Therefore in the second step, we create links from all the dominated *attribute values* to the dominating *attribute values*, where domination is based on the criteria discussed earlier. For example - we observe that the *Resolution* value of the selected product 3, is dominated by the *Resolution* value of product 1. So, a link going out from the *Resolution* value node of product 3 to the *Resolution* value node of product 1 is drawn. Similar process is followed for all the other *attribute values*, with the newly formed links shown in dashed lines in Figure 4. A node should have high utility if it is pointed to by large number of high utility nodes; this circularity has a parallel with the observation that a

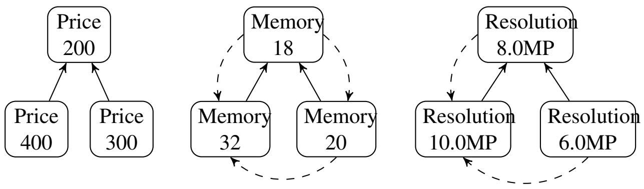


Fig. 4. Links formed due to the local dominance of *attribute values*

web page is important if it is pointed to by several important pages, and is hence resolved using PageRank[16] over the graph in Figure 4. Applying Pagerank on the graph shown in Figure 4 gives us utilities of individual ***attribute values***.

The final step involves recalculating the utility values of all the products in the catalog. Let the *Price*, *Resolution* and *Memory* values of an arbitrary product c be denoted by P_c, R_c and M_c respectively. Similarly, $Price$, *Resolution* and *Memory* values of the selected product(product 3) are denoted by P_3, R_3 and M_3 respectively. Now, the revised utility of a product *c* is calculated as

$$U(c)=sim(P_3,P_c)\cdot util(P_3)+sim(R_3,R_c)\cdot util(R_3)+sim(M_3,M_c)\cdot util(M_3) \quad (5)$$

where $util(P_3)$ is the utility of $Price$ = 200 (product 3's $Price$), calculated by running PageRank[16] on the graph shown in Figure 4. The system then presents to the user, the top k products according to the revised utilities as obtained from Eq. 5. The user can either terminate the interaction if she is satisfied by a product in the current recommendation set or can issue a preference for any one of the shown products and the system re-estimates the utilities of the products through the same process outlined above. The *sim* function used in Eq. 5 is the SimRank based similarity measure between attribute value pairs, calculated according to Equations 1 and 2 applied to a bipartite graph, similar to the one shown in Figure 1. The *util* function used in Eq. 5 is not a predefined weight value for a particular attribute. Instead, it is a dynamic measure of importance of a particular *attribute value*. Individual feature value utility estimation is made efficient in the case of UtilSim by the use of PageRank(applied on a relatively small graph like the one shown in Figure 4), during every recommendation cycle. UtilSim can scale with increasing number of products since SimRank based similarity estimation between products is done offline. Although large number of attributes might lead to efficiency issues, theoretically, we can encode dominance knowledge into the system as a one time effort. However, in practical recommendation settings, it may not always be cognitively appealing to expose users to a system that requires specification of large number of features.

4 Experimental Results

We compare UtilSim with well known preference based approaches - MLT[14], wMLT[14] and MLT-AS[15]. We use two standard datasets in our experiments - **Camera**[1] and **PC**[14]. The **Camera** dataset contains 210 cameras with 10 attributes and the **PC** dataset contains 120 PC's with 8 attributes. We use a leave one out methodology similar to the one used in [15], where each product is removed from the products database and used to generate a particular query of interest. For each query, that product is considered as target, which is most similar to the product from which the query is generated. The target corresponding to a particular query is computed using the weighted similarity model(used by MLT, WMLT and MLT-AS approach) and not the SimRank based notion of similarity on which UtilSim based. Hence, UtilSim starts off with a bit of a disadvantage. We report the average number of cycles and unique products presented to a user en route a target product during a recommendation dialogue, as reported

[1] `http://josquin.cs.depaul.edu/~rburke/research/downloads/camera.zip`

in [15,10]. For both the datasets, we generated queries of length 1, 3 and 5 to distinguish between ***difficult***, ***moderate*** and ***easy*** queries respectively[22]. We refer to queries of length 1, 3 and 5 as Q-1, Q-3 and Q-5 respectively. Q-1 corresponds to a non expert user having limited domain knowledge who finds it difficult to express her preferences clearly at the start whereas Q-5 corresponds to a user close to being an expert, who is able to specify her preferences clearly. A total of 3938 and 2382 queries were generated for the Camera and PC dataset respectively. In all our experiments, 4 products are presented in every recommendation cycle. Also, all the algorithms are made to select that product in every cycle which is most similar to the target. The only difference is that while all other algorithms use the weighted similarity model, UtilSim uses a similarity measure based on SimRank.

Highly Focused Recommendation Framework

For a highly focused recommendation framework, we simulate an artificial user who is relatively sure of her preferences and who, during each cycle, chooses a product which is maximally similar to the target product. As can be seen from Figure 5a through Figure 5d, UtilSim outperforms all the other algorithms, in terms of cycles and unique items. Specifically for the Q-1 case on the PC dataset, while MLT, MLT-AS and wMLT take 11.705 cycles(and 22.14 items), 7.82 cycles(and 16.85 items) and 6.66 cycles(and 19.26 items) to reach the target respectively, UtilSim takes 5.07 cycles(and 15.45 items) to reach the target, a 56% reduction in terms of cycles and 30% reduction in terms of unique items over MLT. A similar trend is observed for the Camera dataset.

Preference Noise

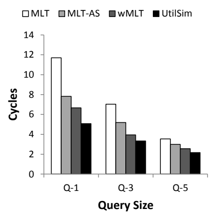
We simulate an agent which does not act optimally during each recommendation cycle by making it choose a product that might not be the most similar to the target. Noise is introduced into the process by disturbing the similarities of the products in the recommendation set to the target product by some random amount within a threshold. We have used a noise level of 5% in our experiments. As explained in [15], preference noise of 5% implies that the similarities of each of the individual products to the target might be changed by up to +/-5% of its actual value. As can be seen from Figure 5e through Figure 5h, UtilSim outperforms the other algorithms on both the datasets. For the Q-1 case on the PC dataset, while MLT, MLT-AS and wMLT take on an average 12.085 cycles(22.72 items), 7.66 cycles(16.55 items) and 8.52 cycles(22.69 items) respectively to reach the target, UtilSim takes 5.25 cycles(16.17 items) to reach the target, a reduction of 56% in terms of cycles and 28% in terms of unique items over MLT. It is interesting to note that wMLT which performs better than MLT-AS in a highly focused framework of recommendation(in terms of number of cycles), does not perform better than MLT-AS in the presence of preference noise(especially for the Camera dataset). This might be due to the fact that in the presence of noise, the recommendation dialogues associated with wMLT include a lot of false leads, whereas MLT-AS is able to neutralize their effect due to its diversity component. For reasons of brevity, it is worth noting that UtilSim's performance is superior to all the other approaches, even at higher levels of noise.

Finding All "good" Items

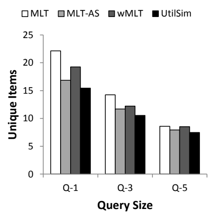
Finding all "good" items is also important for users who only have some vague idea about their preferences, because they can be sure that the recommender system will recommend all the "interesting" products efficiently. Given a predefined target product for a particular query, chosen as described earlier, we also treat as target for that query, those products which are "better" than the target product according to some criterion. For the Camera dataset, a camera which has a ***higher*** *Resolution*, *Optical Zoom*, *Digital Zoom*, *Memory* and ***lower*** *Weight* and *Price* than the target product is also added to the list of target products. For the PC dataset, a PC having ***higher*** *Speed*, *Memory*, *Capacity* and ***lower*** *Price* than the target product is also considered as a target product. Let the set of target products for a query be defined as T. During every recommendation cycle, the simulated agent selects that product from the recommendation set which has the highest average similarity to all the elements in T. If an element from T is part of the recommendation set in a particular cycle, then that element is removed from T and a similar process is followed until T becomes empty. As shown in Figure 5i and Figure 5j, for the Q-1 case on the PC dataset, while MLT, MLT-AS and wMLT take on an average 23.83 cycles(40.87 items), 18.30 cycles(31.28 items) and 15.19 cycles(35.94 items) respectively, UtilSim takes 12.02 cycles(30.73 items) to reach the target product, a reduction of 49% in terms of cycles and 24% in terms of unique items over MLT. For the Camera dataset, Figure 5k and Figure 5l present an interesting case study where UtiSim significantly outperforms MLT-AS in terms of number of cycles but under performs MLT-AS in terms of number of unique items shown. For the Q-1 case on the Camera dataset, MLT-AS shows 36.26 unique items(reduction of 37.66% as compared to MLT), whereas UtilSim shows 38.77 items(reduction of 33.35% as compared to MLT). The marginal increase in the number of unique items notwithstanding, we think that UtilSim is of value in this scenario as well because of the huge reduction it is able to offer in terms of the number of cycles. For Q-1 on Camera dataset, UtilSim takes 14.71 cycles(reduction of 61.79% compared to MLT) as compared to MLT-AS which takes 24.26 cycles(reduction of 36.98% compared to MLT).

Why UtilSim Works

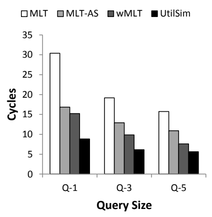
We considered reduced samples of the original PC dataset - the size of the smaller datasets ranging from 30 to 120 products. The number of cycles taken by different algorithms to reach the target are reported in Figure 6. We devised an algorithm, MLT-sRank, which works exactly like the MLT approach, except that it uses a SimRank based similarity measure to estimate utility instead of the weighted similarity model which is used by MLT. As we can see from Figure 6, when the dataset size is between 30 and 90 products, MLT-sRank, which uses SimRank based similarity as a substitute for utility, under-performs MLT. This is because SimRank does not get sufficient data to model similarities in the domain effectively. But once it finds sufficient data(120 products), it, on its own outperforms MLT. More interestingly, we see that for any sample size, UtilSim, with its additional layer of "dominance" knowledge over SimRank based similarity, is able to perform better than MLT-sRank. Moreover, starting from dataset size of 60 onwards, UtilSim starts outperforming even MLT, even though MLT-sRank, based on the same SimRank based similarity as UtilSim, cannot match up to MLT.



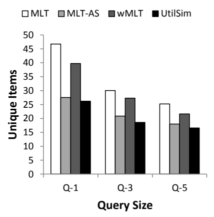
(a) PC Data



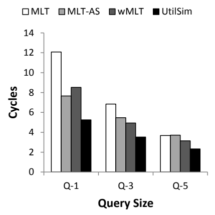
(b) PC Data



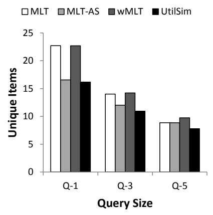
(c) Camera Data



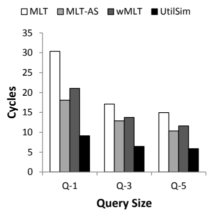
(d) Camera Data



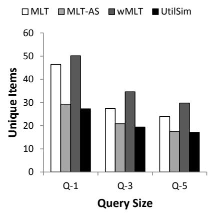
(e) PC Data



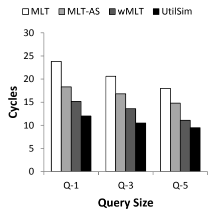
(f) PC Data



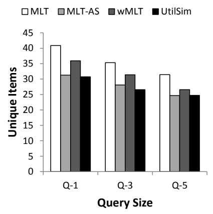
(g) Camera Data



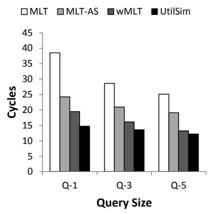
(h) Camera Data



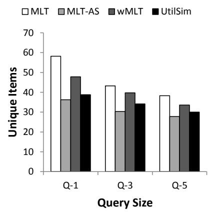
(i) PC Data



(j) PC Data



(k) Camera Data



(l) Camera Data

Fig. 5. Performance Analysis for (a) Highly focussed scenario (Row 1) (b) Preference Noise (Row 2) (c) ”All Good Items task” (Row 3)

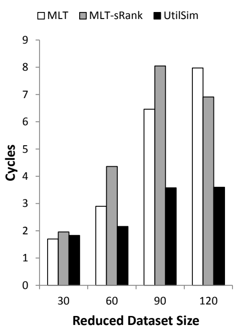


Fig. 6. Cycles for variable number of products

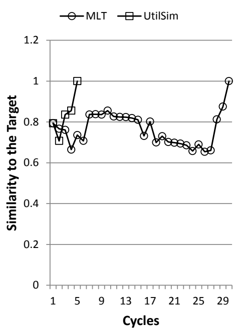


Fig. 7. Selected products’ similarity to target in a session

In Figure 7, we have plotted the similarity(corresponding to weighted similarity model) of the preferred product in every cycle, to the target product during a typical recommendation dialogue. Generally, we would expect the similarity of selected products to the target, to increase over a period of a few cycles, instead, for MLT, we observe that it encounters similarity troughs from cycles 1-4 and from 8-27, during which the similarity to the target changes a little or falls down. We quantify this ability of each algorithm to lead to the target by aggregating the slopes of all the lines joining the successively preferred products divided by the number of cycles taken by the algorithm for a typical recommendation dialogue like the one shown in Figure 7. We then calculate the average of such scores obtained across all the recommendation dialogues for the PC dataset. The scores achieved by UtilSim across Q-1, Q-3, Q-5 are 0.045, 0.029, 0.016 respectively. The scores achieved by MLT for Q-1, Q-3, Q-5 are 0.026, 0.020, 0.011 respectively. The higher scores achieved by UtilSim across dialogues associated with all query sizes show that it has a better ability to lead the user to the target.

5 Conclusions

In this paper, we proposed a novel algorithm, UtilSim, which helps non-expert/novice users (who have limited knowledge about product features and have difficulty in expressing their preferences clearly) discover their preferences in an iterative and adaptive fashion. UtilSim leads to efficient recommendations by combining domain-specific "dominance" knowledge with SimRank based similarity as opposed to weighted similarity which is generally used in case based recommender systems. The promising results obtained by the use of SimRank based similarity has positive implications for domains where it might be difficult to define local similarity measures across attributes. We observe that the utility function used in UtilSim can get richer by taking into account feature interactions. Most Preference-Based Feedback algorithms do not model feature interactions and we would like to pursue this line of research in the future.

References

1. Pu, P., Chen, L.: User-Involved Preference Elicitation for Product Search and Recommender Systems. Ai Magazine 29, 93–103 (2008)
2. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Recommender Systems Handbook, pp. 1–35 (2011)
3. Smyth, B.: Case-Based Recommendation. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) Adaptive Web 2007. LNCS, vol. 4321, pp. 342–376. Springer, Heidelberg (2007)
4. Shimazu, H.: Expertclerk: navigating shoppers' buying process with the combination of asking and proposing. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence, IJCAI 2001, vol. 2, pp. 1443–1448. Morgan Kaufmann Publishers Inc., San Francisco (2001)
5. Smyth, B., Cotter, P.: A Personalized TV Listings Service for the Digital TV Age. Knowledge-Based Systems 13(2-3), 53–59 (2000)
6. Burke, R., Hammond, K., Yound, B.: The findme approach to assisted browsing. IEEE Expert 12(4), 32–40 (1997)

7. McCarthy, K., Reilly, J., McGinty, L., Smyth, B.: Experiments in dynamic critiquing. In: Proceedings of the 10th International Conference on Intelligent User Interfaces, IUI 2005, pp. 175–182. ACM, New York (2005)
8. Reilly, J., Zhang, J., McGinty, L., Pu, P., Smyth, B.: A comparison of two compound critiquing systems. In: Proceedings of the 12th International Conference on Intelligent user Interfaces, IUI 2007, pp. 317–320. ACM, New York (2007)
9. Zhang, J., Jones, N., Pu, P.: A visual interface for critiquing-based recommender systems. In: Proceedings of the 9th ACM Conference on Electronic Commerce, EC 2008, pp. 230–239. ACM, New York (2008)
10. Llorente, M.S., Guerrero, S.E.: Increasing retrieval quality in conversational recommenders. IEEE Trans. Knowl. Data Eng. 24(10), 1876–1888 (2012)
11. Bridge, D., Ferguson, A.: An expressive query language for product recommender systems. Artif. Intell. Rev. 18(3-4), 269–307 (2002)
12. Jeh, G., Widom, J.: Simrank: a measure of structural-context similarity. In: KDD 2002: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 538–543. ACM Press, New York (2002)
13. Mcsherry, D.: Similarity and compromise. In: Ashley, K.D., Bridge, D.G. (eds.) ICCBR 2003. LNCS (LNAI), vol. 2689, pp. 291–305. Springer, Heidelberg (2003)
14. McGinty, L., Smyth, B.: Comparison-based recommendation. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS (LNAI), vol. 2416, pp. 575–589. Springer, Heidelberg (2002)
15. Smyth, B., Mcginty, L.: The power of suggestion. In: IJCAI, pp. 127–132. Morgan Kauffman (2003)
16. Lawrence, P., Sergey, B., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University (1998)
17. Gori, M., Pucci, A.: Itemrank: a random-walk based scoring algorithm for recommender engines. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence, IJCAI 2007, pp. 2766–2771. Morgan Kaufmann Publishers Inc., San Francisco (2007)
18. Teppan, E.C., Felfernig, A.: Calculating decoy items in utility-based recommendation. In: Chien, B.-C., Hong, T.-P., Chen, S.-M., Ali, M. (eds.) IEA/AIE 2009. LNCS (LNAI), vol. 5579, pp. 183–192. Springer, Heidelberg (2009)
19. Dan, A., Thomas, W.: Seeking Subjective Dominance in Multidimensional Space: An Explanation of the Asymmetric Dominance Effect. Organizational Behavior and Human Decision Processes 63(3), 223–232 (1995)
20. Simonson, I.: Choice Based on Reasons: The Case of Attraction and Compromise Effects. Journal of Consumer Research 16(2), 158–174 (1989)
21. Knijnenburg, B.P., Willemsen, M.C.: Understanding the effect of adaptive preference elicitation methods on user satisfaction of a recommender system. In: Proceedings of the Third ACM Conference on Recommender Systems, RecSys 2009, pp. 381–384. ACM, New York (2009)
22. Salamó, M., Reilly, J., McGinty, L., Smyth, B.: Knowledge discovery from user preferences in conversational recommendation. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 228–239. Springer, Heidelberg (2005)