Acquiring User Profiles from Implicit Feedback in a Conversational Recommender System

Henry Blanco
Faculty of Computer Science
Free University of Bozen-Bolzano
Bozen-Bolzano, Italy
University of Oriente
Santiago de Cuba, Cuba
Henry.BlancoLores@stud-inf.unibz.it

Francesco Ricci
Faculty of Computer Science
Free University of Bozen-Bolzano
Bozen-Bolzano, Italy
fricci@unibz.it

ABSTRACT

Query revisions in a conversational system can be efficiently computed by assuming that the profiles of the potential users are in a predefined, a priori known and finite set. However, without any additional knowledge of the actual profiles distribution, the system may miss the true profiles of the users, hence deteriorating the system performance. We propose a method for identifying a tailored set of profiles that is acquired by analysing the implicitly shown preferences of the users that interacted with the system. We show that with the proposed method the system can efficiently identify good query revisions.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*

Keywords

Recommender systems; conversational systems; profile learning; utility function

1. INTRODUCTION

Preference elicitation methods based on implicit feedback are used in Recommender Systems (RS) to deal with users lacking the motivation to completely specify their preferences up front and to help them to unveil preferences that they may not be even aware of [5]. These methods acquire the user model by observing how the user interacts with the RS and are typically implemented in conversational systems. This means that, in a series of user-system interactions, the system acquires and revises the user preferences by monitoring the user reactions to the information or the recommendations shown [5].

We introduced in [2] a conversational technique, which was further improved in [1], aimed at helping users to revise their queries when searching in a product catalog. The products are described by their features and the system suggests query revisions that are likely to retrieve products which maximize the user utility. For example, in our model if the user submits the query "I want a hotel with AC and parking", the system, rather than immediately retrieving the products that satisfy this query, may suggest some query revisions such as: "are you interested also in a sauna; or in a tennis court?". User preferences are then inferred by comparing the query revision chosen by the user, among those that were suggested. For instance, in the above example, the user may additionally request hotels with "sauna" and not "tennis court". The system generated query revisions are those estimated to retrieve products with the largest user's utility, according to the inferred preferences.

In [1] we overcame the computational complexity of finding good query revisions, which is described in [2], by assuming that the users can be modelled with a large but predefined finite set of profiles. Here a profile is a vector of weights which conveys the importance the user assigns to each product feature. The predefined profiles represent the possible users that the system believes it may interact with.

In this paper we reduce the computation time needed to identify the best query revisions by considering a better representative set of profiles: the system iteratively adds to the considered set of profiles those estimated to belong to the users that approached the system. We call this process "acquiring user profiles" because the users will not explicitly reveal their preferences and it is up to the system to conjecture the definition of their profiles by observing their user-system interactions. We show that the proposed method does identify good query revisions and the computation time, compared to the case when the same number of uniformly distributed profiles are used, is reduced.

The paper is organized as follows. Related work is discussed in Section 2. Section 3 describes our models for representing products, queries, user preferences and the unfolding of the user-system interaction. The experiment design and the results are shown in Section 4. Finally the conclusions and future work are described in Section 5.

2. RELATED WORK

In recommendation approaches where items are described by their features, the user profile, which models the user's preferences, is typically a vector of weights measuring the importance assigned by the user to each feature or feature value. While in content-based approaches the user profile is generally built by mining the users ratings [4] (explicit feedback), in our approach the user profile is constructed by observing a series of user selections among alternative system suggested queries, which is a kind of implicit feedback.

A popular approach where feature-based user preferences are estimated using the user's choices in products comparisons, rather than ratings to items, is Conjoint Analysis [6]. Here the estimation of the user profile depends on several factors: the utility model for each feature (e.g. linear, not linear), the products presented to stimulate the elicitation of the user preferences, the method for estimating the user utility function (e.g., multiple regression analysis).

The major problems are here generated by the large number of products that the user is required to compare and the consequent complex estimation of the true user utility function. In our work we have simplified the preferences elicitation process by iteratively acquiring from the complete user profiles space a small number of profiles compatible with the current user's inferred preferences. This is done only if the system has not yet acquired a profile that is compatible with the current user's (implicitly acquired) preferences.

In Analytic Hierarchy Processing, a popular multi-criteria decision making technique, the feature importance weights are acquired by asking the user to perform pair-wise comparisons between product features [3]. Even in this case the main problem is the large number of pairwise comparisons to be made by the user. Conversely the number of comparisons that the user is required to make in our approach is significantly smaller because only the queries with the largest estimated utility are suggested.

Finally, in the Compound Critiquing approach the importance of the product features (weights) is also used to suggest the best products to the user [7]. Here the feature importance weights are modified according to the user critiques to the selected products. But these profiles are not preserved along the various user-system interactions, making impossible the successive analysis on the users' preferences. Conversely in our approach we maintain this information and we are able to elicit the true preferences' distribution of the users.

3. MODELS AND ALGORITHM

3.1 Queries and Products

A product is modeled by an n-dimensional Boolean vector $p = (p_1, \ldots, p_n)$ with n attributes. Boolean product features, such as Air Conditioning, are modeled as Boolean attributes, where $p_i = 1$ ($p_i = 0$) means that the product has (not) the i-th feature. Moreover, using Boolean attributes it is also possible to model discrete features by representing each particular discrete feature value with a single Boolean attribute [1]. A catalogue is a set of products $C = \{p^{(1)}, \ldots, p^{(k)}\}$.

Likewise, queries are represented with Boolean vectors $q = (q_1, \ldots, q_n)$, where $q_i = 1$ means that the user desires products having the i-th attribute and $q_i = 0$ means that the user has not yet declared her interest on that feature.

A query is *satisfiable* if there exists a product in the catalogue C such that all the features expressed in the query as desired ($q_i = 1$) are present in that product. More details and examples can be found in [1].

3.2 Query Editing Operators

We consider a scenario where the system, with an appropriate GUI, suggests query revisions to the user. These query revisions are generated using application specific editing operators, as those listed here and considered in this study:

- $add_1(q, i)$, $i \in idx0(q)$;
- $trade_{1,2}(q, i, j, k)$, $i \in idx1(q)$ and $j, k \in idx0(q)$;
- $add_2(q, i, j)$, $i, j \in idx0(q)$;
- $trade_{1,3}(q, i, j, k, t)$, $i \in idx1(q)$ and $j, k, t \in idx0(q)$.

$idx0(q)$ and $idx1(q)$ are the sets of indexes corresponding to not requested and requested features in q respectively. The operators add_x extend the query q by adding one or two features not yet requested. Similarly, the operators $trade_{x,y}$ extend the query q by considering two or three not yet requested features in detriment of one already requested. The new queries generated by these operators can be presented to the user, however, the system's goal is not to suggest all the possible query revisions, but only those that retrieve products with the largest utility for the user.

3.3 User Utility Function

A user utility function, or user profile, is defined by a vector of weights $w = (w_1, \ldots, w_n)$, $0 \leq w_i \leq 1$. w_i models the importance that the user assigns to the i-th feature of a product: $w_i = 0$ means that the user has no desire for the i-th feature; if $w_i > w_j$ then the i-th feature is more important than the j-th one; and if $w_i \geq w_j$ then the i-th feature is at least as important as the j-th one. If $w_i = w_j$ then the user is indifferent between these two features. The utility of the product $p = (p_1, \ldots, p_n)$ is as follows:

$$U_w(p) = \sum_{i=1}^{n} w_i \times f_i(p) \tag{1}$$

where $f_i(p)$ is the value function of the i-th feature of p. Moreover, the utility of the query q for a user with profile w, $U_w(q)$, is defined as the utility $U_w(p)$ of a product p with the same definition as q, i.e., $q = p$.

3.4 User-System Interaction

The considered user-system interaction is described in Figure 1. Here we assume that: 1) the system does not explicitly ask the user about her preferences; 2) the users are rational and prefer products with larger utility; 3) the system contemplates a finite number of predefined profiles P, that is initially small (just 5 in our experiments) and it is expanded anytime the system does not find in P a profile compatible with the current user's implicitly expressed preferences.

A user-system interaction evolves in cycles. At the beginning of a cycle (steps 3 and 4) the user either submits the initial query (first cycle) or selects one query revision in the *AdviseSet* (successive cycles). By observing the user selection the system infers some constraints Φ on the user utility function (step 5). That is, if q_s is the selected query, then the system infers the constraints $U_w(q_s) \geq U_w(q)$, for all the suggested queries $q \in AdviseSet$. Note that here w is unknown. If the set P_Φ of the system's contemplated profiles in P that are compatible with the inferred constraints Φ is not empty then the candidate query revisions are generated by using the query editing operators discussed in Section 3.2 and the queries that are not $satisfiable$ are discarded

```
1.  $\Phi = \emptyset$, $P_\Phi = P$ acquired profiles, $AdviceSet = \emptyset$
2.  do {
3.    Present the $AdviceSet$ to the user;
4.    $q_s$ = initial query or one in the $AdviceSet$;
5.    Infer constraints analysing $q_s$, and add them to $\Phi$;
6.    Identify $P_\Phi$, the compatible profiles in $P$;
7.    if $P_\Phi \neq \emptyset$
8.      Generate satisfiable $CandidateQueries$;
9.      $AdviceSet = NotDominated(CandidateQueries)$;
10.     $AdviceSet = TopKUtility(AdviseSet)$;
11.   else
12.     Generate compatible profiles and add them to $P$
13. } while ($AdviceSet \neq$ null and user wants advice)
```

Figure 1: User-System interaction process.

(step 8). In order to suggest the query revisions that are more likely to increase the user utility, the system first discards the queries that are proved to be *dominated* (step 9) and then select the top-K queries with the largest expected utility.

A query $q \in CandidateQueries$ is considered *dominated* if there exists another query $q' \in CandidateQueries$ such that for all the weight vectors $w \in P_\Phi$: $U_w(q') > U_w(q)$. Finally the top K queries (5 in our experiments) with the largest expected utility are identified and included in the *AdviseSet* to the user (step 10). The expected utility of a query is computed as $E[q] = \frac{1}{|P_\Phi|}\sum_{w \in P_\Phi} U_w(q)$.

Otherwise, if there is no profile in P compatible with the inferred constraints in Φ (else statement) then there is a failure point and the system generates a small set of profiles, which are compatible with Φ, and adds them to P (more information on this step is provided in the next section).

This cycle is repeated until the user does not want more advices or there are no more query suggestions. Note that when a new user approaches the system this interaction is repeated and eventually new profiles are added to P, making P larger and larger as needed.

3.5 Acquiring User Profiles

As mentioned in Section 3.4 the absence of compatible profiles is a failure point and prevents the computation of the query revisions for the user. At this point the system conjectures that the current users's profile is missing and adds some profiles that satisfy the inferred constraints. We perform this step by first generating some random profiles $w = (w_1, \ldots, w_n)$ taken from the system conjectured distribution of the user profiles (uniform in our experiments, i.e., $w_i = rand([0,1])$). Then, the generated profiles are added to P, the system considered profiles, if they satisfy the inferred constraints in Φ. In this situation we say that the system has "acquired" the current user profile. These generated compatible profiles update the system knowledge of the users. In our experiments we have considered scenarios where 2, 10 or 20 profiles are generated and acquired when the system does not find in P a compatible profile while interacting with a user. We decided to add more than one single compatible profile to speed up the acquisition process and because these generated profiles represent only hypotheses on the true user's profile.

4. EXPERIMENTS AND RESULTS

In order to validate the proposed user profile acquisition technique and double check that it is not jeopardising the system ability to suggest good query revisions, as shown in [1], we conducted several experiments where 500 virtual users interacted with our system according to the user-system interaction described in Section 3.4.

In order to understand how much difficult it is the profile acquisition task and to identify the best number of profiles that should be included at each interaction failure point, we simulated different scenarios where the users approaching the system were clustered into 1, 5 or 10 groups and where the number of "acquired" profiles at each failure point was 2, 10 or 20. We simulated users in a group with similar preferences by generating their profiles using a normal distribution around a group's prototype profile: for each group, a group's prototype $w^* = (w_1, ..., w_n)$ was randomly set and the other profiles in the group were generated by enforcing that $w = w^* \times \mathcal{N}(1, \sigma)$, where $\mathcal{N}(1, \sigma)$ is the normal distribution with average 1 and standard deviation σ. We used two values for σ, 0.1 and 0.05, to model groups with more or less disperse members. The vectors of weights were then normalized to sum to 1.

We mainly examined the impact of the number of acquired profiles and the quality of the system suggestions, measured with the average utility shortfall. The utility shortfall is the difference in utility between the query selected by the user from the *AdviceSet* and the query that retrieves the products with the largest true user's utility. We used one product database comprising 4056 hotels in "Trentino" (Italy), each of them described by 11 features [1].

We have found that the larger is the number of different user groups the larger is the number of profiles that the system must add to P while interacting with the users in order to provide the same quality of query suggestions, i.e., a utility shortfall smaller than 0.05. This can be observed in columns 4 and 6 of Table 1. In order to achieve this level of performance the system adds more profiles when interacting with more disperse groups of users ($\sigma = 0.1$) compared to when interacting with groups of users with less diverse preferences ($\sigma = 0.05$). Moreover, this conclusion is also supported by the fact that for a given number of acquired profiles (e.g., 45 or 150, as in columns 3 and 5) the quality of query suggestions decreases when interacting with an increasing number of user groups. That is, the task of acquiring the user profiles becomes harder when interacting with more heterogeneous users.

Furthermore, we have discovered that acquiring a larger number of profiles at each failure point (10 or 20) is not effective, especially when there are more groups of diverse users. (see columns 3 and 5 of Table 1). This is better illustrated in Figure 2 that is showing the case when the system interacted with 10 groups of users with dispersion $\sigma = 0.1$. Adding less profiles at each failure point lets the system to converge faster to low utility shortfall values. Furthermore, it is shown that acquiring a selected set of profiles, using the proposed method, compares favourably to an approach where a uniformly distributed set of profiles (of the same cardinality) is used ("no learning").

Finally, the computation time for generating the query suggestions for a user when P contains 100 profiles is around 22 milliseconds. This is roughly 4 times smaller than what is required for computing the query suggestions (86 mil-

Table 1: Utility Shortfall (USh) for different sizes of the acquired profiles set and number of acquired profiles to obtain a desired USh level

		$\sigma = 0.05$		$\sigma = 0.1$	
# of groups	# of profiles acquired at a failure point	USh with 45 added profiles	# profiles for USh ≤ 0.05	USh with 150 added profiles	# profiles for USh ≤ 0.05
1	2	0.050	44	0.050	150
	10	0.052	45	0.076	320
	20	0.054	50	0.116	400
5	2	0.077	66	0.069	170
	10	0.311	140	0.126	280
	20	0.370	220	0.216	460
10	2	0.120	72	0.077	200
	10	0.371	200	0.139	420
	20	0.399	300	0.264	650

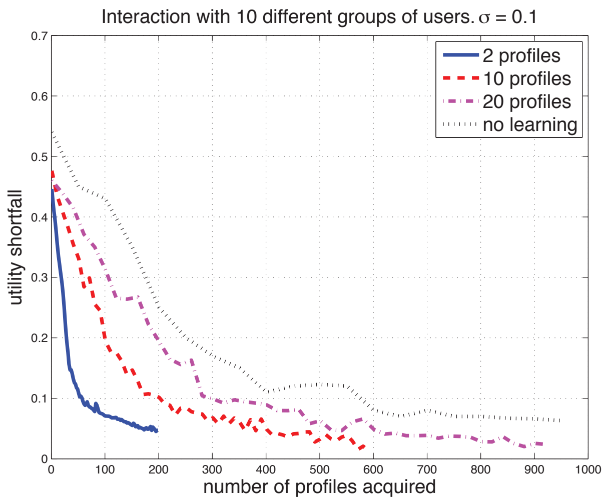

Figure 2: Average utility shortfall in simulated interactions - 10 groups of users, $\sigma = 0.1$.

liseconds) when considering one thousand randomly sampled profiles, which is the number of profiles necessary to achieve the same level of performance of our proposed method with 100 profiles (see Figure 2).

5. CONCLUSIONS AND FUTURE WORK

In this paper we have introduced a method for identifying a tailored set of profiles that represents the users that have interacted and may interact in the future with a conversational system.

We have shown that acquiring this tailored set of profiles is harder when the users are clustered in several groups with a large diversity of their preferences. Additionally, we have shown that acquiring few well selected profiles every time the system fails to generate query suggestions makes the system to converge fast to better performances. We have also shown that the proposed approach substantially improves a simpler method where the profiles are added using the system known a priori distribution of the users preferences. It is worth noting that using the proposed method the system can suggest good query revisions just considering few but tailored profiles and the system response time in multiple and concurrent interactions is in the order of the few seconds per user, making it possible to use it in realistic web scenarios.

Nevertheless, there are some important issues that must be addressed in future work such as initialising the system with a better a priori distribution of user profiles, and devising a solution for removing profiles that may not be anymore suited to represent the users approaching the system, i.e., supporting the dynamic change of the distribution of the users' preferences.

6. REFERENCES

[1] H. Blanco and F. Ricci. Inferring user utility for query revision recommendation. In *Proceedings of the 28th Symposium On Applied Computing*, volume 1, pages 245–252. ACM, 2013.

[2] D. Bridge and F. Ricci. Supporting product selection with query editing recommendations. In *Proceedings of the 2007 ACM conference on Recommender systems*, RecSys '07, pages 65–72. ACM, 2007.

[3] D.-N. Chen, P. J.-H. Hu, Y.-R. Kuo, and T.-P. Liang. A web-based personalized recommendation system for mobile phone selection: Design, implementation, and evaluation. *Expert Syst. Appl.*, 37(12):8201–8210, 2010.

[4] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer Verlag, 2011.

[5] L. McGinty and J. Reilly. On the evolution of critiquing recommenders. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 419–453. Springer Verlag, 2011.

[6] M. Scholz. From consumer preferences towards buying decisions. In *21st Bled eConference eCollaboration: Overcoming Boundaries Through Multi-Channel Interaction*, pages 223–235, 2008.

[7] J. Zhang and P. Pu. A comparative study of compound critique generation in conversational recommender systems. In *Procs. of 4th Intl. Conf. on Adaptive Hypermedia & Adaptive Web-Based Systems*, pages 234–243. Springer-Verlag, 2006.