Query Personalization based on User Preferences

Georgia Koutrika[1]

koutrika@di.uoa.gr

Abstract. Query Personalization is the process of dynamically enhancing a query with related user preferences stored in a user profile with the aim of providing personalized answers. The underlying idea is that different users may find different things relevant to a search due to different preferences. Essential ingredients of query personalization are: (a) a model for representing and storing preferences in user profiles, and (b) algorithms for the generation of personalized answers using stored preferences.

1. Introduction

A user accessing an information system with the intention of satisfying an information need, may have to reformulate the query issued several times and sift through many results until a satisfactory, if any, answer is obtained. This is a very common experience especially for Web searchers, due to information abundance and users' heterogeneity in the Web. A critical observation is that "*different users may find different things relevant when searching*" because of different preferences, goals etc. Thus, they may expect different answers to the same query. Consider a simple case: two users, Al and Julie, access a web-based movies database both searching for comedies. Al is a fan of director W. Allen, while Julie is not. Most systems would consider only the request issued and return to both users the same, exhaustive list of comedies. However, storing user preferences in user profiles gives a system the opportunity to return more focused, personalized (and hopefully smaller) answers. Fig. 1 illustrates the difference between traditional and personalized search.

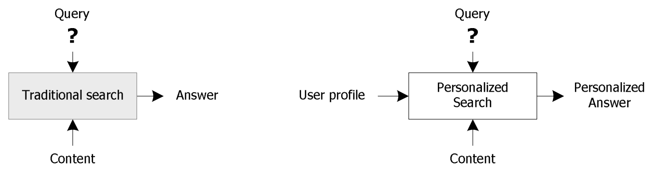


Fig. 1. Traditional vs. personalized search

Based on the above, this dissertation proposes *Query Personalization*, i.e. a process of dynamically enhancing a query with related user preferences stored in a user profile with the purpose of providing focused customized answers. Focusing on the

[1] Supervisor: Prof. Yannis Ioannidis

user enables a shift from what is called ‘consensus relevancy’ where the computed relevancy for the entire population is presumed relevant for each user, toward ‘personal relevancy’ where relevancy is computed based on each individual’s characteristics. Personalized results for Al would include W. Allen’s comedies, while personalized results for Julie would not.

Given a query and a profile, a personalized answer is built by specifying: *(a)* the number K of top preferences from the user profile that should affect it, and *(b)* the number L (L≤K) of those preferences that should at least be satisfied. Parameters K and L can be specified directly by the user or derived based on various criteria on the query context, such as user location, time, device, etc. Query personalization proceeds as follows (Fig. 2): (Preference Selection) Top K preferences are derived from the user profile. (Personalized Answer Generation) These are combined with the query, and a personalized answer is returned satisfying L of the K preferences.

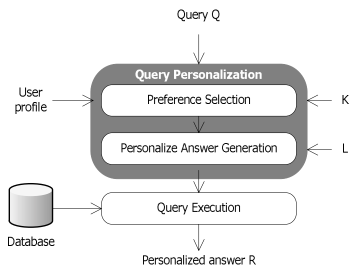


Fig. 2. Query Personalization Architecture

Contributions. The contributions of this work are the following:

- *Preference model*. User preferences are stored as degrees of interest in atomic query elements (selection and join conditions), which may be used to transform a query. The degree of interest expresses the interest of a person to include the associated condition into the qualification of a query. Specific logic is introduced for derivation of preferences combining stored atomic ones. This model combines expressivity and concision and provides a direct way to personalize queries.
- *Preference Selection*. Preference selection from a structured user profile is formulated as a graph computation problem. Efficient algorithms that derive the top K preferences from a user profile that are related to a query are described. We consider preferences that are syntactically related to or conflicting with a query taking into account the schema of the underlying database.
- *Generation of Personalized Answer*. The top K preferences derived from the user profile may be integrated into the initial query so that the resulting one should return results satisfying at least L from the top K preferences. Alternative query rewriting schemes have been examined. A specialized algorithm that allows for progressive generation of personalized results is also proposed.
- *Optimization*. Query personalization is formulated as a constrained optimization problem, where constraints may concern the personalized query execution time and result size. This is the first realistic approach to personalization that takes into

account real-time factors and conditions in order to return personalized results. Constrained query personalization is formulated as a state space search problem. Each personalized query that is a potential solution is a state in a space (a node in a graph), characterized by degree of interest, execution cost, and result size. Transitions (edges) arrange states in partial orders based on each of the aforementioned query parameters. State-space search algorithms are devised that take advantage of these partial orders to solve these problems efficiently.

– *Experimental Evaluation.* Experiments have demonstrated the efficiency of the proposed algorithms, providing insight as to the appropriateness of the preference model, and showing the benefits of query personalization.

2. Related Work

Content personalization. Several approaches to content personalization have been developed by the IR community, most of them falling into two major categories: information filtering and recommendation systems.

Information filtering systems aim at satisfying long-term information needs of people. A long-term information need is represented as a query stored in the system that is continuously executed over dynamic data with the purpose of returning new or updated relevant information to the user. A stored query or set of queries comprise a user profile based on which an information filtering system collects and distributes relevant information [19, 8].

Recommendation systems produce predictions, recommendations, opinions that help a user to evaluate and select objects [20, 21]. The basic idea is that a user selects objects, for instance by marking them, or querying them, and the system identifies other similar objects, based on which it produces recommendations or predictions regarding what the user would like.

A search process could take into account not only the query issued but also the characteristics of the user submitting this query. These could be stored in a user profile. This observation gave rise to the idea of *personalized search*. Early efforts have originated from the field of Information Retrieval [22] and have mainly conceived personalized search as a problem of re-ranking the results of a query based on a user profile [18, 23].

The survey of related work has revealed two important facts [15]. First, the vast majority of existing content personalization approaches concern unstructured data. Second, there are only a few proposals regarding personalized searching, most of them defining the problem as one of result re-ranking according to a user profile rather than as a query modification one. These facts underline the value of this dissertation that studies the problem of personalization of queries over databases based on user preferences stored in a profile.

Preference Modelling. Preference is a fundamental notion in areas of applied mathematics, philosophy, and computer science that deal with decisions and choice. In Mathematical Decision Theory, preferences (or utilities) model economic behaviour [7]. In Philosophy, they are used to reason about values, desires, and duties

[9]. In AI, they capture agents' goals [6]. In Databases, they are used for the formulation of 'soft' query criteria. We distinguish two categories:

– *Qualitative* approaches aim at a relative formulation of preferences, such as a user prefers comedies over westerns [2, 17, 5, 12]. This formulation is natural for a human and results in partial orders of results. However, absolute specification of preference significance is not possible.
– *Quantitative* approaches aim at an absolute formulation of preferences, such as a user likes comedies very much and westerns to a lesser degree [1, 10, 3, 4, 11]. This allows for total ordering of results and the straightforward selection of those matching user preferences.

The approaches above have focused on the expression of preferences at the query level and on algorithms aiming at the efficient execution of these queries. Our approach aims at the representation and storage of user preferences in profiles. It allows for the expression of several preference types and provides a direct way to personalize queries.

3. User Preference Model

Our approach is applicable to any graph model representing information at the level of entities and relationships. Preferences may be expressed for values of attributes, and for relationships between entities. Preferences for relationships indicate to what degree, if any, entities related are influenced by each other (i.e. by preferences on each other). Here, we briefly overview the main characteristics of the proposed user preference model. Details on the full-fledged model can be found in [13].

Atomic Preferences. Given our focus on personalization of queries, our preference model assigns degrees of interest to query constructs, which may then be used to transform, i.e. personalize a query. Preferences for values of attributes are stored as atomic selections (*atomic selection preferences*) and preferences for relationships between entities are stored as atomic join conditions (*atomic join preferences*). The latter indicate to what degree related entities are mutually influenced by preferences and they are directed, in the sense that they indicate how preferences on the right-hand-side join relation influences the left-hand-side join relation.

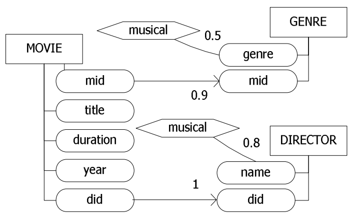
Example. Consider the following relations, which are a subset of a database schema about movies:

```
MOVIE(mid, title, year, duration, did)
DIRECTOR(did, name), GENRE(mid, genre)
```

Fig. 3(a) shows an example user profile. Degree of interest equal to `0` indicates lack of any interest in the condition, while degree equal to `1` indicates extreme ('must-have') interest.

A user's preferences over a database's contents are expressed on top of the *personalization graph*. This is a directed graph `G(V, E)` that is an extension of the database schema graph. Nodes in `V` are (*a*) *relation nodes*, one for each relation in the schema, (*b*) *attribute nodes*, one for each attribute of each relation in the schema, and (*c*) *value nodes*, one for each value that is of any interest to this user. Edges in `E` are (*a*) *selection edges*, from an attribute node to a value node representing a potential selection condition, and (*b*) *join edges*, from an attribute node to another attribute node representing a potential join condition between these attributes. Fig. 3(b) shows the personalization graph corresponding to the user profile of Fig. 3(a).

p_1: doi(GENRE.genre='musical') = 0.5
p_2: doi(MOVIE.mid=GENRE.mid) = 0.9
p_3: doi(MOVIE.did=DIRECTOR.did) = 1.0
p_4: doi(DIRECTOR.name='W. Allen') = 0.8



(a)profile (b)personalization graph

Fig. 3. User preferences

Implicit Preferences. By composing atomic user preferences on conditions (edges) that are adjacent in the personalization graph, one obtains implicit preferences, i.e. preferences on complex query elements that are conjunctions of atomic ones (directed acyclic paths in G). In particular, if p is an implicit preference containing m atomic preferences p_i, then $p = p_1 \wedge \ldots \wedge p_m$. For example, p_3 and p_4 are composed into the following implicit preference for movies directed by W. Allen:

MOVIE.did = DIRECTOR.did **and** DIRECTOR.name = 'W. Allen'

The degree of interest in an implicit preference p is a function $f_{\otimes}$ of the degrees of interest in the constituent atomic ones and must be non-increasing as the length of the corresponding directed path increases:

$$\text{doi}(p)=f_{\otimes}(d_1, \ldots d_m) \leq \mathit{min}(\{d_1, \ldots d_m\}), \quad d_i = \text{doi}(p_i) \qquad (1)$$

Combinations of Preferences. Given a set of user preferences, whether atomic or implicit, one may form logical combinations of them. *Ranking functions* are proposed for estimating user interest in logical combinations of preferences. One may see three different philosophies.

– *Inflationary*. The degree of interest in multiple preferences satisfied together increases with the number of these preferences expressing a philosophy of 'the more the better'. An example function is the following:

$$r_1 = 1 - \prod_{i=1}^{N}(1 - d_i) \qquad (2)$$

– *Dominant*. The degree of interest in multiple preferences satisfied together is exactly equal to the degree of the most interesting of these preferences, i.e.

$$r_2 = \max(\{d_1, d_2, \ldots d_N\}) \qquad (3)$$

– *Reserved*. The degree of interest in multiple preferences satisfied together is between the highest and the lowest degrees of interest among the original preferences. The underlying principle is that the degree of interest in satisfying multiple preferences should primarily depend on the importance of them. The following function belongs to this category:

$$r_3 = 1 - \prod_{i=1}^{N}(1 - d_i)^{1/N} \qquad (4)$$

The appropriateness of a ranking function is judged only by the philosophy of the approach taken towards personalization and, more importantly, by how closely it reflects human behaviour. We have experimented with the above functions, and the results provide insight as to the appropriateness and intuitiveness of each one of them.

4. Preference Selection

The first step of the query personalization process deals with for the extraction of the top K preferences related to a query. A basic question to be answered by a system is which preferences are considered related to a query. Our approach is to take into account the schema of the underlying database in order to identify preferences syntactically related to or conflicting with a query. A preference is syntactically related to a query, if the corresponding path on the personalization graph is attached to a query relation. E.g., an implicit preference related to a query about movies is:

```
MOVIE.mid=GENRE.mid and GENRE.genre='comedy'
```

Parameter K is specified with the use of some criterion, e.g., it may specify that the top 5 preferences should be selected, or that the desired degree of interest should be greater than 0.8. Preference selection from a structured user profile is formulated as a graph computation problem. Efficient preference selection algorithms perform a *best-first traversal* of the personalization graph in order to construct paths in order of decreasing importance. These are described in [13] and [16].

5. Personalized Answer Generation

The top K preferences derived from the user profile may be integrated into the initial query so that the resulting one should return results satisfying at least L from the top K preferences. Among several possible query rewritings one could use to personalize a query with a set of preferences [16], we illustrate the most efficient one through an example: a user, whose profile is shown in Fig.3, issues a query about movies:

```
select      title from     MOVIE
```

Assume that the following preferences have been selected by the system for inclusion in the query:

```
MOVIE.did = DIRECTOR.did and DIRECTOR.name = 'W. Allen'
MOVIE.mid = GENRE.did and GENRE.genre = 'musical'
```

First, a set of sub-queries is constructed, each one separately integrating one of these preferences into the original query.

```
Q1: select title from     MOVIE M, DIRECTOR D
    where  M.did = D.did and D.name = 'W. Allen'
Q2: select title from     MOVIE M, GENRE G
    where  M.mid=G.mid and G.genre='musical'
```

The final query is built as the union of these sub-queries:

```
select      title  from    Q1 Union All Q2
group by    title  having   count(*)= 2
```

Finally, the resulting query is passed on the query optimizer of the underlying database system, where it is executed. The results of this query may be ranked based on their degree of interest.

Still, this query rewriting method has shortcomings, the most important being that results are returned only after they have all been retrieved, merged, grouped and ordered. There is no way to progressively output results. Therefore, we have provided an algorithm for progressive generation of personalized results [13].

6. Constrained Query Personalization

In principle, query personalization is an *optimization problem*: given a query q posed by a user u, its goal is to identify the parts of the profile of u that, when combined with q, would *maximize* the interest of u in the results of q. In practice, this problem statement may lead to unrealistic solutions, since maximum interest is achieved by incorporating all preferences of u into q, but the resulting “over-personalized” query is likely to be very expensive or have an empty answer. Taking into account execution time and result size leads to a redefinition of query personalization as a *constrained optimization problem*, where constraints are expressed as an upper bound on execution time of the final query and/or a lower or upper bound on its result size. Under this more general point of view, one realizes that query personalization does not necessarily imply optimization of user interest, but could also be, for example, optimization of execution time under constraints on user interest.

The family of constrained optimization problems that arise in the spirit discussed above is referred to as *Constrained Query Personalization* (*CQP*) [14]. Depending on the parameter being optimized and the constraints placed on the others, different answers will be delivered even to the same user issuing the same query. Each time the correct CQP problem is determined by several real-time factors comprising the search context, such as the device used, the network connection, or even some transient user requirements of the moment.

6.1 State Space Search

Constrained query personalization is formulated as a state space search problem. Let C be a set of preferences related to a given query Q. Each state in a CQP problem corresponds to a query built by integrating a subset of preferences into the initial query, i.e. $Q_x := Q \wedge C_x$, where $C_x \subseteq C$. Query parameters comprise the features of the corresponding state. Transitions are based on syntactic modifications to a state with known implications (increase/decrease) on state parameters and they arrange states in partial orders based on each of the aforementioned query parameters. We define two categories of transitions, cost-based and degree-based. Each category creates a different state space (same nodes, different edges). State-space search algorithms are devised that take advantage of these partial orders to solve these problems efficiently.

CQP problems have similar formulation, query parameter properties and partial orders derivable from syntactic transformations of personalized queries. These correspondences enable us to treat them in a very similar way. Therefore, in order to sketch our approach, we focus on one CQP problem, i.e. maximize degree of interest with an upper cost bound (e.g. cost < 190). We define two cost-based transitions:

– The *Horizontal* neighbour of a state is derived by inserting the preference from `C` that immediately follows the lowest cost preference of the current state, and it has higher cost and higher degree of interest than the current state.
– *Vertical* neighbours of a state are derived by replacing a preference in the state by its successor from `C` provided that the latter is not already in the state. Vertical neighbours are ordered in decreasing cost.

Fig. 4 shows the state space for $C = \{c_1, c_2, c_3, c_4, c_5\}$. Vertical transitions are depicted in dashed lines, Horizontal ones in solid lines. Numbers show the cost of the query produced by adding the corresponding preferences into the initial query.

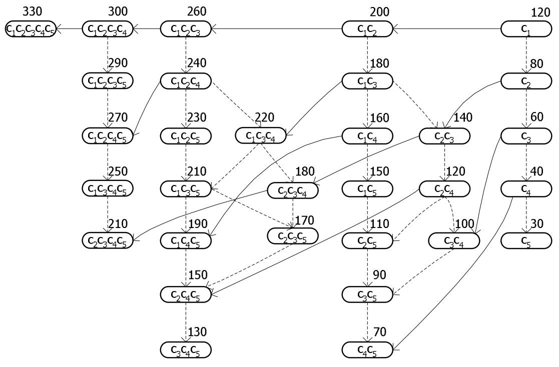


Fig. 4. Cost State space

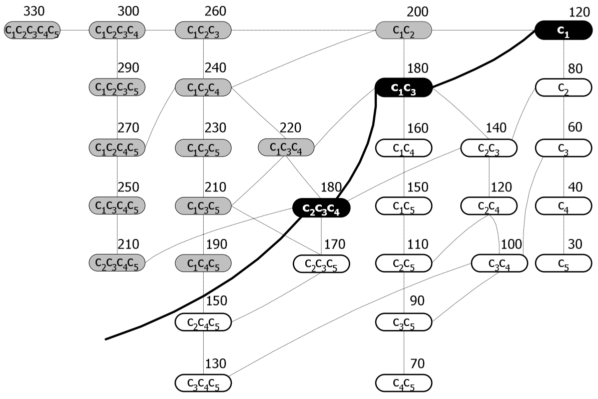


Fig. 5. Basic idea of state space search

The basic idea in state space search is finding a set of nodes that are not reachable from each other and satisfy the cost constraint, while their parents do not. These nodes are called *boundaries*. Boundaries on every group form a virtual borderline that partitions the cost state space into two sets of nodes as Fig. 5 shows: those satisfying the cost constraint (cost < 190), and those not. Then, the solution to the CQP problem considered is a node of maximum degree of interest belonging to the first set.

We have devised algorithms that are based on cost-based or degree-based transitions and may be used for solving any kind of CQP problem due to the fact that all CQP problems are quite similar to each other, both in terms of their formulation but also in terms of characteristic properties of the query parameters that are involved in them. These algorithms are provided in [14].

7. Conclusions

This dissertation proposes personalization of database queries, a process of dynamically enhancing a query with preferences stored in a user profile with the purpose of generating a customized, focused, and hopefully smaller answer for a specific user. We have presented a preference model that combines expressivity and concision. User preferences are stored as degrees of interest in atomic query elements (individual selection and join conditions), which may be used to transform a query. We proposed a personalization framework according to which personalized results should satisfy at least L out of the top K preferences from the user profile that are related to the given query. We formulated preference selection from a structured user profile as a graph computation problem and we provided efficient preference selection algorithms. We studied several alternative methods to rewrite the initial query into a personalized one that integrates the preferences selected from a user profile and returns results satisfying L from the top K preferences selected. We proposed an efficient algorithm for progressive retrieval of personalized results. Furthermore, we have formulated query personalization as a constrained optimization problem, where constraints may concern the personalized query execution time and result size. Constrained query personalization is formulated as a state space search problem. We devised state-space search algorithms that take advantage of these partial orders to solve these problems efficiently. Finally, this dissertation includes experimental results demonstrating the efficiency of our algorithms, providing insight as to the appropriateness of the proposed preference model, and showing the benefits of query personalization.

Several research issues are still left open and are the subject of our ongoing and future work. We plan to study how our preference model may express preference dependencies. Another challenging research direction is towards user models that combine multiple user aspects, e.g. preferences, abilities, demographic data etc. Models of increased expressive power such as those outlined above require advanced query personalization mechanisms and logic. For example, combining and reconciling information stored in diverse profiles, and profile hierarchies are challenging topics that need to be addressed. Finally, we are very interested in methods for the automatic construction of user profiles based on the preference model described in this work. Existing methods have mainly focused on construction of simple keyword profiles. The complexity of database queries, however, compared to keyword-based searches render most of the existing techniques useless. Fresh ideas are required to come up with effective approaches in this environment.

8. References

1. Agrawal, R., Wimmers, E. A (2000). Framework for Expressing and Combining Preferences. Proc. Of the ACM Int'l Conf. on Management of Data.
2. Borzsonyi, S., Kossmann, D., Stocker, K. (2001). The Skyline Operator. Proc. Of Intl. Conf. On Data Engineering, 421–430.
3. Bruno, N., Chaudhuri, S., Gravano, L. (2002). Top- k Selection Queries over Relational Databases: Mapping Strategies and Performance Evaluation. ACM TODS, 27(2), 153-187.
4. Chang, K., Hwang, S. (2002). Minimal Probing: Supporting Expensive Predicates for Top-k Queries. Proc. Of the ACM Int'l Conf. on Management of Data.
5. Chomicki, J. (2003). Preference Formulas in Relational Queries. ACM TODS, 28(4), 427–466.
6. Delgrande, J., Schaub, T., Tompits, H. (2000). Logic Programs with Compiled Preferences. Proc. Of the EU. Conf. On AI.
7. Fishburn, P. (1999). Preference Structures and Their Numerical Representations. Theor. Comput. Sci. 217, 359–383.
8. Foltz, P., and Dumais, S. (1992). Personalized Information Delivery: An Analysis of Information Filtering Methods. Comm. Of the ACM, 35(12), 51-60.
9. Hansson, S. O. (2001). Preference Logic. In Handbook of Philosophical Logic, D. Gabbay, Ed. Vol. 8
10. Hristidis, V. Koudas, N. Papakonstantinou, Y. (2001). PREFER: A System for the Efficient Execution of Multiparametric Ranked Queries. Proc. Of the ACM Int'l Conf. on Management of Data
11. Ilyas, I, Aref W., Elmagarmid, A. (2003). Supporting Top-k Join Queries in Relational Databases. Proc. Of the Int'l VLDB Conf.
12. Kießling, W. (2002). Foundations of preferences in database systems. Proc. Of the 28th Intl. VLDB Conf..
13. Koutrika, G., Ioannidis, Y. (2005). Personalized Queries under a Generalized Preference Model. Proc. of 21st ICDE, 841-852, 5-8 April 2005, Tokyo, Japan.
14. Koutrika, G., Ioannidis, Y. (2005). Constrained Optimalities in Query Personalization. In Proc. of ACM SIGMOD, 73-84, 13-16 June 2005, Baltimore, Maryland, USA.
15. Y. Ioannidis, G. Koutrika. Tutorial: Personalized Systems: Models and Methods from an IR and DB Perspective. 31st Intl. Conf. VLDB, August 30 – September 2, 2005, Norway
16. Koutrika, G., Ioannidis, Y. (2004). Personalization of Queries in Database systems. In Proceedings of 20th Intl. Conf. ICDE, 597-608, 30 March - 2 April 2004, Boston, MA, USA.
17. Lacroix, M., Lavency, P. (1987). Preferences: Putting More Knowledge into Queries. Proc. of Int'l VLDB Conf., 217–225
18. Liu, F. Yu, C. Meng, W. (2004). Personalized Web Search for Improving Retrieval Effectiveness IEEE TKDE 16(1) Jan. 2004.
19. Mooney, R., Roy, L. (2000). Content-Based Book Recommending Using Learning for Text Categorization. Proc. of the 5h ACM Conf. on Digital Libraries, 195–204
20. Resnick, P. Varian, H. R. (1997). Recommender systems. Comm. of the ACM, 40(3), 56-58, Mar 1997
21. Schafer, J. B., Konstan, J. A., Riedl, J. (2002). Meta-recommendation Systems: User-controlled Integration of Diverse Recommendations. Proc. of the 11th CIKM.
22. Sieg, A., Mobasher, B., Lytinen, S., Burke, R. (2003). Concept Based Query Enhancement in the ARCH Search Agent. International Conference on Internet Computing: 613-619
23. Tanudjaja, F., Mui, L. (2002). Persona: A Contextualized and Personalized Web Search. Proc. of the 35th Hawaii Int'l Conf. on System Sciences