

# Personalization of Structured Queries with Personal and Collaborative Preferences

Georgia Koutrika<sup>1</sup>

**Abstract.** The underlying idea of personalized searches is that different people may find different things relevant; therefore, they may expect different answers to the same query. One of the primary ways to personalize a search is query personalization. This is the process of dynamically enhancing a query with related preferences stored in a user profile with the purpose of providing more focused answers. However, in many cases, this may be too restrictive, since users are always provided with items matching their own profile. In every day life, people often make choices not only based on their personal preferences but also taking into account preferences, opinions or choices of other people.

Tapping into the preferences and choices of others enables people to examine alternative or additional options and make more informed decisions. In this work, we introduce collaborative concepts in query personalization and we define collaborative query personalization. This is the process of dynamically enhancing user queries by taking into account personal as well as collaborative preferences, i.e. preferences of like-minded or trusted people. We discuss the implementation of a prototype collaborative query personalization system for structured queries including preliminary results of experimental evaluation.

## 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 [33]. Storing user preferences in user profiles gives a system the opportunity to personalize web searches.

One of the primary ways to personalize a search for an active searcher is query personalization. Query personalization is the process of dynamically enhancing a query with related preferences

stored in a user profile with the purpose of providing more focused personalized (and hopefully smaller) answers. However, in many cases, this may be too restrictive since users are always provided with items that match their own profile. In every day life, people often make choices not only based on their personal preferences but also taking into account preferences, opinions or choices of other people. For example, when Ann is out shopping and searching for a dress, she might also want to have a look at dresses her friend Mary would consider. Likewise, regarding comedies, she is a fan of director W. Allen but also appreciates Joanne's taste on comedies; thus, when searching for comedies, she might also like to have a look at comedies that would be presented to Joanne based on her profile. Consequently, when searching, people may not be solely interested in answers matching their personal interests and preferences, but they might also be interested in answers other people would receive from the system for the same query. Tapping into the preferences and choices of others enables people to examine alternative or additional options and essentially make more informed decisions. Based on this observation, *collaborative query personalization* is the process of dynamically enhancing user queries by taking into account personal as well as collaborative preferences, i.e. preferences of like-minded people.

*Contributions.* In this paper, we describe collaborative query personalization in the context of database queries. We identify the major steps in this process and we provide algorithms for their implementation along with initial experimental results.

*Outline.* Section 2 presents related work. Section 3 provides background information on user preference modelling and query personalization in the context of structured queries. Section 4 describes collaborative query personalization phases. Section 5 presents experimental results. Section 6 discusses future work.

## 2. RELATED WORK

*Information Filtering* systems filter their responses on the basis of a rudimentary user profile storing long-term user interests [10]. There are several systems for: Usenet news (SIFT [34]), documents (InRoute [6]), music (Lyrictime [21]), and so forth.

*Collaborative Filtering* systems have automated the everyday procedure of relying on recommendations from other people whenever personal experience of the alternatives is not sufficient

---

<sup>1</sup> University of Athens, Athens, Greece, email: koutrika@di.uoa.gr

Partially supported by the Information Society Technologies (IST) Program of the European Commission as part of the DELOS Network of Excellence on Digital Libraries (Contract G038-507618)

for making choices [27]. Two large categories of approaches are distinguished: memory-based and model-based. Memory-based algorithms predict the rating of an item based on the ratings of users who are similar to the active user. There are a number of similarity measures, e.g., Pearson Correlation [26], Constrained Pearson Correlation [28], Spearman Correlation [13]. Predictions of how much a user will like an item are usually computed by taking the weighted average of the opinions of a set of nearest neighbors for that item. Model-based algorithms use an aggregated model learnt offline [3]. Information Filtering and Collaborative Filtering systems are Recommender Systems. Other types of Recommender Systems include Knowledge-based [31] and Hybrid systems [1, 29].

*Personalized Search* systems rely on the idea that different people may find different things relevant; therefore, they may expect different answers to the same query. The primary ways to personalize a search are: *personalized results ranking* and *query personalization*. The former is achieved by re-ranking the results returned by the query according to the user profile (METIORE [5]), or by building on graph analysis algorithms (Persona [32], [15, 23]). Query personalization is mainly augmentation of the query with terms (ARCH [30], Outride [24], Inquirus [12], [20]) or structures ([17, 18, 19]) stored in the user profile.

*Collaborative Web search* uses the past search behaviour (queries and selections) of a community of users to promote search results that are relevant to the community. Similarity of queries may be judged based on their results [11, 35] or the terms involved in them [9]. Then, one approach is to use similar past queries to automatically expand new queries [9]. The idea is that top documents returned by the query from a pool of documents are also top documents returned by similar queries and are good source for automatic query expansions. Another approach is to re-rank results of a query [4, 7] or recommend relevant results [2, 25] based on results selected for other similar queries. Commercial collaborative search engines have become to emerge, e.g., Jookster! [16], My Web 2.0 [22], Eurekster [8], Jeteye [14].

*Comparison.* We adopt the preference model presented in [19] and we extend the query personalization framework presented there by introducing concepts of memory-based collaborative filtering. In contrast to Collaborative Filtering, we make use of structured profiles instead of a user-ratings matrix or vectors of features to correlate users. Moreover, people are correlated based on their preferences that are relevant to a given query rather than based on their entire profile. Furthermore, a Collaborative Filtering engine takes a list of items from a user and provides a list of predicted ratings for these items, or it returns the top predicted items for the active user from the database. Collaborative Query Personalization finds the top preferences of similar users. On the other hand, Collaborative web search introduces Collaborative Filtering ideas in Information Retrieval. We introduce collaborative ideas in Query Personalization in order to enrich the returned results, matching personal preferences, with additional results satisfying collaborative preferences. Collaborative Search approaches compute similarity of queries (based on their terms or selections); we compute similarity between profiles, and we do not mine recommendations from past queries or selections, instead we identify preferences from profiles of similar users, which are used to enhance a query.

### 3. BACKGROUND

In this section, we provide background information on preference modelling for structured queries and query personalization.

#### 3.1 User preference model

Any personalization effort requires a model for the representation of preferences stored in user profiles. In this work, we adopt an existing user preference model for structured profiles [19], whose basic constituents are described below. For our examples, we consider a movies database described by the following relations:

```

THEATRE(tid, name, phone, region, ticket),
PLAY(tid, mid, date), GENRE(mid, genre),
MOVIE(mid, title, year, duration),
CAST(mid, aid, role), ACTOR(aid, name),
DIRECTED(mid, did), DIRECTOR(did, name)

```

**Atomic Preferences.** Preferences are stored in profiles at the level of atomic query elements (atomic selection and join conditions). An atomic preference for a condition  $q$  is expressed by the *degree of interest* in  $q$ ,  $\text{doi}(q)$ , which is a real number in the range  $[0, 1]$ .  $\text{doi}(q) = 0$  indicates lack of any interest in the condition, while  $\text{doi}(q) = 1$  indicates extreme interest. Atomic selection preferences indicate user interest in attribute values. Atomic join preferences indicate to what degree related entities are mutually influenced by preferences. These are directed, in the sense that they indicate how preferences on the right-hand-side join relation influence the left-hand-side join relation.

Figure 1 presents preferences stored in a user profile for a movie database. This user likes comedies very much ( $p_1$ ) and adventures to a lesser degree ( $p_2$ ). Her favorite actress is N. Kidman ( $p_3$ ). She is moderately interested in director D. Lynch ( $p_4$ ). These preferences are expressed as degrees of interest in specific atomic selections. Moreover, she has preferences expressed over joins between relations of the schema, to allow queries on one to take into account her preferences on the others. For example, preferences  $p_5$  and  $p_6$  express user interest in the director and the genre of a movie, respectively. The user is more interested in the director than in the genre of a movie.

$p_1$	GENRE.genre = 'comedy'	0.9
$p_2$	GENRE.genre = 'adventure'	0.7
$p_3$	ACTOR.name = 'N. Kidman'	0.8
$p_4$	DIRECTOR.name = 'D. Lynch'	0.7
$p_5$	MOVIE.mid = DIRECTED.mid	0.9
	DIRECTED.did = DIRECTOR.did	1.0
$p_6$	MOVIE.mid = GENRE.mid	0.8

Figure 1. Example preferences from a user profile

A particular user's preferences over the contents of a database can be expressed on top of the *personalization graph* of the database. This is a directed graph  $G(\mathbf{v}, \mathbf{e})$  ( $\mathbf{v}$  is the set of nodes and  $\mathbf{e}$  is the set of edges) that is an extension of the database schema graph. Nodes in  $\mathbf{v}$  are: (a) *relation nodes*, one for each relation in the database schema; (b) *attribute nodes*, one for each attribute of each relation; (c) *value nodes*, one for each value of each attribute of each relation that is of any interest to a particular user. Edges in  $\mathbf{e}$  are: (a) *selection edges*, from an attribute node to

a value node representing the potential selection condition connecting the corresponding attribute and value; (b) *join edges*, from an attribute node to another attribute node representing the potential join condition between the corresponding attributes. Degrees of interest are indicated as labels on the graph's edges.

**Implicit Preferences.** By composing atomic 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  $\mathcal{G}$ ). In particular, if  $p$  is an implicit preference comprised of  $m$  atomic preferences  $p_i$ , i.e.  $p = p_1 \text{ and } \dots \text{ and } p_m$ , its degree of interest is a function  $f_{\otimes}$  of the degrees of interest in the constituent atomic ones:

$$\text{doi}(p) = f_{\otimes}(\text{doi}(p_1), \dots, \text{doi}(p_m)) \quad (1)$$

The degree of interest in an implicit preference must decrease as the length of the corresponding directed path increases:

$$f_{\otimes}(d_1, \dots, d_m) \leq \min(\{d_1, \dots, d_m\}), \quad d_i = \text{doi}(p_i) \quad (2)$$

For instance, preferences  $p_4, p_6$  are composed into a preference for movies directed by D. Lynch:

```
MOVIE.mid = DIRECTED.mid and
DIRECTED.did = DIRECTOR.did and
DIRECTOR.name = 'D. Lynch'
```

Using multiplication as function  $f_{\otimes}$ , the degree of interest in the example preference above is  $0.9 \times 1.0 \times 0.7 = 0.63$ .

**Conjunctions of Preferences.** The degree of interest in the conjunction of a set of selection preferences is a function of the degrees of interest in them. Then, the results of a personalized query can be ranked based on the preferences they satisfy.

### 3.2 Query personalization

Based on the query personalization framework presented in [19], given a query and a profile, a personalized answer is built by specifying: (a) the number  $\kappa$  of top preferences from the user profile that should affect it, and (b) the number  $L$  ( $L \leq \kappa$ ) of those preferences that should at least be satisfied. In other words, a personalized answer comprises a set of results of the initial user query that satisfy any  $L$  of a user's top  $\kappa$  preferences. Parameters  $\kappa$  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. For instance, a criterion for  $\kappa$  could specify that the top four user preferences should be considered, and a criterion for  $L$  could specify that the personalized results returned should satisfy at least two of these preferences.

Query personalization proceeds in two phases: (*Preference Selection*) the top  $\kappa$  preferences that are related to a given query are derived from the user profile; (*Personalized Answer Generation*) these and the original user query are combined into a query that qualifies results satisfying  $L$  of the top  $\kappa$  user preferences. This query is executed instead of the original one.

## 4. COLLABORATIVE QUERY PERSONALIZATION

*Collaborative Query Personalization* is the process of dynamically enhancing a user query with personal and

collaborative preferences, i.e. preferences of like-minded people, which are related to this query. The objective of collaborative query personalization is to allow a user to tap into results other users would obtain for the same query along with results that match one's personal profile. People are correlated based on their preferences for a given query rather than based on their entire profile. For example, when searching for comedies, Ann's taste may be closer to Joanne's, whereas when querying about theatres, her preferences are more similar to John's. Therefore, for queries of the first type, she might also like results that Joanne would receive, while for queries on theatres, she would probably prefer theatres according to John's preferences.

Building upon the query personalization framework presented in [19], given a query and a user profile, the effect of collaborative query personalization is determined by the following parameters:

- ( $\kappa, L$ ):  $\kappa$  is the number of top preferences from this user's profile that are related to the query and should affect the personalized answer, and  $L$  ( $L \leq \kappa$ ) is the number of those preferences that should at least be satisfied.
- ( $\kappa_c, L_c$ ):  $\kappa_c$  is the number of top collaborative preferences that are related to the query and should affect the personalized answer, and  $L_c$  ( $L_c \leq \kappa_c$ ) is the number of those preferences that should at least be satisfied.

Then, the system answer should include results satisfying at least  $L$  of the top  $\kappa$  preferences of the active user and results satisfying at least  $L_c$  of the top  $\kappa_c$  collaborative preferences. Parameters  $\kappa_c$  and  $L_c$  may be determined with the use of several criteria similar to those discussed for their counterparts  $\kappa$  and  $L$ . Their default values should be  $\kappa_c = \kappa$  and  $L_c = L$  capturing the intuition that users perceive results in the same way, regardless of what kind of preferences they satisfy. Thus, results should satisfy collaborative preferences according to the same criteria as results satisfying personal preferences.

Given a query  $Q$ , a user profile  $U$  and criteria for the specification of parameters  $\kappa, L$ , and  $\kappa_c, L_c$ , collaborative query personalization proceeds as follows:

*Personal Preference Selection.* Preferences from the active user's profile that are related to the query  $Q$  are derived.

*Neighbour Identification.* Users are correlated based on their preferences for the given query rather than based on their entire profile, and the users that are most similar to the active searcher are identified as neighbours.

*Collaborative Preference Selection.* The top  $\kappa_c$  collaborative preferences are computed based on the neighbours' preferences that are related to the query  $Q$ . These should be different from the top  $\kappa$  personal preferences from the active user's profile that will be also used to personalize the query.

*Personalized Answer Generation.* The system combines query  $Q$ , personal and collaborative preferences and produces results satisfying at least  $L$  of the top  $\kappa$  preferences of the user and results satisfying at least  $L_c$  of the top  $\kappa_c$  collaborative preferences.

Figure 2 depicts a high-level architecture of a collaborative query personalization system.

**Example.** We will use the following scenario as a running example. Assume that a user submits the following simple query, and expects results satisfying  $L = 2$  of her top  $\kappa = 5$  preferences and results satisfying  $L_c = 1$  of  $\kappa_c = 2$  collaborative ones.:

```
select title from MOVIE
```

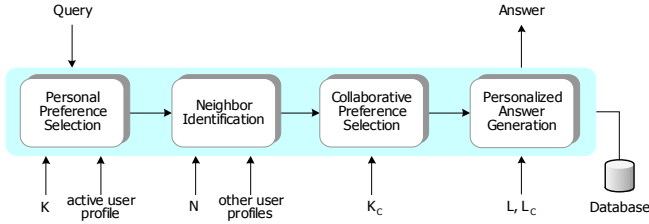


Figure 2. System Architecture

## 4.1 Personal preference selection

Given a query  $Q$  and the active user's profile  $U$ , the first phase of collaborative query personalization derives the set  $P$  of atomic and implicit selection preferences from  $U$  that are related to  $Q$ . A preference may be related to a query at a semantic or syntactic level. For example, a preference for director W. Allen is semantically related to a query about comedies, whereas a preference for director M. Tarkowski is not. In order to determine semantic relationships, additional knowledge about data stored in the database is required. We deal with syntactic relationships, i.e. relationships dictated by the database schema. A preference is syntactically related to a query  $Q$ , if its path on the personalization graph is attached to a relation in  $Q$ . For example, a preference related to the user query above is:

MOVIE.mid = GENRE.mid **and** GENRE.genre = 'drama'

The preference selection algorithm PPS is adopted from [19]. It exploits the property of decreasing degree of interest as the length of the path of a preference increases (Formula 2), and performs a best-first traversal of the personalization graph corresponding to  $U$  to extract preferences from  $U$  that are related to  $Q$  in decreasing order of degree of interest.

**Example.** Assume that the personal preferences related to the user query are these:

```

MOVIE.mid = GENRE.mid
and GENRE.genre = 'comedy'           0.72
MOVIE.mid = DIRECTED.mid
and DIRECTED.did = DIRECTOR.did
and DIRECTOR.name = 'D. Lynch'       0.63
MOVIE.mid = CAST.mid
and CAST.aid = ACTOR.aid
and ACTOR.name = 'A. Hopkins'        0.595
MOVIE.mid = CAST.mid
and CAST.aid = ACTOR.aid
and ACTOR.name = 'N. Kidman'         0.56
MOVIE.mid = GENRE.mid
and GENRE.genre = 'adventure'        0.56
  
```

## 4.2 Neighbour identification

The objective of collaborative query personalization is to allow a user to tap into results other like-minded users would obtain for the same query, apart from results matching one's personal profile. People are correlated based on their preferences for the current query rather than based on their entire profile. The purpose of the second phase of collaborative query personalization is to identify the closest neighbours of the active user with respect to the current query and, for each one of them, derive their

preferences that are related to the current query  $Q$ . Similarity of each user to the active one is computed based on their common preferences that are related to  $Q$ . Two preferences are *common*, if they involve the same atomic or implicit condition but possibly different degrees of interest.

For this purpose, the algorithm NEI, provided in Figure 3, is used. Its inputs are: a query  $Q$ , the set  $P$  of personal preferences from the active user's profile that are related to  $Q$ , and a set  $U$  of user profiles in the system. These could be all user profiles stored in the system or a cluster of them, provided that off-line clustering of users has been performed in advance in order to separate users that have nothing in common. The algorithm computes the similarity of each user to the active one using as a similarity measure the Pearson correlation coefficient [26].

For each one of the  $N$  most similar users, the algorithm outputs their similarity weight,  $w_i$ , with respect to the active user and the current query, as well as the set  $P_i$  of preferences related to the current query  $Q$  that are not common with the top  $K$  ones of the active user. Removal of common preferences from the set of preferences returned for each neighbor is the simplest way to ensure that the set of collaborative preferences finally produced is different from the set of personal preferences for the active user and the given query. Other alternatives are possible; for presentation purposes, they are not discussed here.

---

### NEI Algorithm

---

```

Input: query  $Q$ ,
          set of personal preferences  $P$ ,
          set of other user profiles  $U$ 
Output: similarity weights  $\{w_i \mid i=1..N\}$ ,
          sets of preferences  $\{P_i \mid i=1..N\}$ 

Begin
1. Foreach user profile  $U_i$  in  $U$ 
   1.1. apply PPS and construct  $P_i$ 
   End for
2. Foreach  $P_i$ 
   2.1.  $P'_i = \{\}$ 
   End for
3. Foreach  $p_j$  in  $P$ 
   3.1. Foreach  $P_i$ 
       3.1.1. If exists  $p_k$  in  $P_i$  common with  $p_j$ 
            $P'_i := \text{add}(P'_i, \{d_j, d_k\})$ 
            $P_i := \text{remove}(P_i, p_k)$ 
       End if
   End for
4. For each  $P_i$  not satisfying criteria
   4.1. Discard  $P'_i$  and  $P_i$ 
   End for
5. Foreach  $P'_i$ 
   5.1. Compute  $w_i$ 
   End for
6. output the top  $N$   $w_i$  and corresponding  $P_i$ 
End
  
```

---

Figure 3. Neighbour Identification

In detail, initially, for each user in  $U$ , the algorithm builds the set  $P_i$  of preferences of that user that are related to query  $Q$ .

P <sub>1</sub>	P <sub>11</sub>	MOVIE.mid = GENRE.mid and GENRE.genre = 'comedy'	0.9
	P <sub>12</sub>	MOVIE.mid = DIRECTED.mid and DIRECTED.did = DIRECTOR.did and DIRECTOR.name = 'D. Lynch'	0.85
	P <sub>13</sub>	MOVIE.year > 1990	0.8
	P <sub>14</sub>	MOVIE.mid = DIRECTED.mid and DIRECTED.did = DIRECTOR.did and DIRECTOR.name = 'W. Allen'	0.78
	P <sub>15</sub>	MOVIE.mid = GENRE.mid and GENRE.genre = 'adventure'	0.5
P <sub>2</sub>	P <sub>21</sub>	MOVIE.mid = PLAY.mid and PLAY.tid = THEATRE.tid and THEATRE.region = 'downtown'	0.95
	P <sub>22</sub>	MOVIE.mid = GENRE.mid and GENRE.genre = 'comedy'	0.8
	P <sub>23</sub>	MOVIE.mid = CAST.mid and CAST.aid = ACTOR.aid and ACTOR.name = 'J. Roberts'	0.8
	P <sub>24</sub>	MOVIE.mid = PLAY.mid and PLAY.tid = THEATRE.tid and THEATRE.ticket < '6 Euros'	0.7
	P <sub>25</sub>	MOVIE.mid = CAST.mid and CAST.aid = ACTOR.aid and ACTOR.name = 'N. Kidman'	0.56
P <sub>3</sub>	P <sub>31</sub>	MOVIE.mid = DIRECTED.mid and DIRECTED.did = DIRECTOR.did and DIRECTOR.name = 'D. Lynch'	0.92
	P <sub>32</sub>	MOVIE.mid = CAST.mid and CAST.aid = ACTOR.aid and ACTOR.name = 'N. Kidman'	0.89
	P <sub>33</sub>	MOVIE.mid = DIRECTED.mid and DIRECTED.did = DIRECTOR.did and DIRECTOR.name = 'W. Allen'	0.88
	P <sub>34</sub>	MOVIE.mid = CAST.mid and CAST.aid = ACTOR.aid and ACTOR.name = 'R. De Niro'	0.75
	P <sub>35</sub>	MOVIE.mid = CAST.mid and CAST.aid = ACTOR.aid and ACTOR.name = 'A. Hopkins'	0.7
P <sub>4</sub>	P <sub>41</sub>	MOVIE.mid = GENRE.mid and GENRE.genre = 'comedy'	0.9
	P <sub>42</sub>	MOVIE.mid = CAST.mid and CAST.aid = ACTOR.aid and ACTOR.name = 'A. Hopkins'	0.85
	P <sub>43</sub>	MOVIE.mid = GENRE.mid and GENRE.genre = 'adventure'	0.7
	P <sub>44</sub>	MOVIE.mid = DIRECTED.mid and DIRECTED.did = DIRECTOR.did and DIRECTOR.name = 'D. Lynch'	0.65
	P <sub>45</sub>	MOVIE.mid = CAST.mid and CAST.aid = ACTOR.aid and ACTOR.name = 'N. Kidman'	0.56
P <sub>5</sub>	P <sub>51</sub>	MOVIE.mid = GENRE.mid and GENRE.genre = 'thriller'	0.85
	P <sub>52</sub>	MOVIE.mid = GENRE.mid and GENRE.genre = 'comedy'	0.85
	P <sub>53</sub>	MOVIE.mid = DIRECTED.mid and DIRECTED.did = DIRECTOR.did and DIRECTOR.name = 'D. Lynch'	0.8
	P <sub>54</sub>	MOVIE.mid = DIRECTED.mid and DIRECTED.did = DIRECTOR.did and DIRECTOR.name = 'W. Allen'	0.7
	P <sub>55</sub>	MOVIE.mid = CAST.mid and CAST.aid = ACTOR.aid and ACTOR.name = 'A. Hopkins'	0.65

Figure 4. Preferences selected from other user profiles

For this purpose, it applies the preference selection algorithm PPS. Subsequently, in order to compute the similarity of each

user in  $\mathcal{U}$  to the active one, their common preferences with respect to  $\mathcal{Q}$  need to be identified. For this purpose, for each set of preferences  $P_i$ , the algorithm builds a set  $P'_i$  of pairs of degrees of interest. A pair  $\{d_j, d_k\}$  in  $P'_i$  corresponds to a common preference of  $P_i$  and  $P$ , where  $d_j$  is its degree of interest in  $P$  and  $d_k$  is its degree of interest in  $P_i$ . In the same time, all preferences in  $P_i$  that are common with preferences in  $P$  are removed. We use a hashing scheme to find common preferences.

Not all preference sets  $P_i$  are considered. If all preferences from a set  $P_i$  are common with preferences in  $P$ , then  $P_i$  and its respective  $P'_i$  are discarded, since the corresponding user has no preferences to recommend to the active one. Moreover, it is important not to trust correlations based on very few common preferences. Therefore, we use the following heuristic in order to select the users that will be compared to the active one: if a user has fewer than  $M$  ( $M \leq$  number of preferences in  $P$ ) common preferences with the active user, it is discarded. This parameter filters the users that will be finally compared to the active one. If it draws near the number of preferences in  $P$ , then a small number of users will be compared. This increases system performance, but, on the other hand, it shrinks the set of preferences from which the top  $K_c$  collaborative ones will be drawn. If  $M$  is very small comparatively to the number of preferences in  $P$ , then many users will be compared to the active one, even if they are not bound to be among the top  $N$  neighbours. Based on the above, as a rule of thumb, we have taken  $M$  to be equal to half the number of preferences in  $P$ .

Finally, the algorithm computes the similarity weight  $w_i$  of any remaining  $P_i$  to  $P$  based on the pairs of common preferences recorded in the corresponding  $P'_i$ . For this purpose, the Pearson correlation coefficient is written, for our purposes as follows:

$$w_i = \frac{\sum_{(d_j, d_k) \in P'_i} (d_j - \bar{d}) (d_k - \bar{d}_i)}{\sqrt{\sum_{d_j \in P'_i} (d_j - \bar{d})^2 \sum_{d_k \in P'_i} (d_k - \bar{d}_i)^2}}$$

where each pair  $\{d_j, d_k\}$  in  $P'_i$  contains the degrees of interest of a common preference,  $d_j$  is the degree of interest of the active user and  $d_k$  is the degree of interest of the other user. In addition,  $\bar{d}$  is the average degree of interest of the active user, and  $\bar{d}_i$  is the average degree of interest of the other user. The algorithm outputs only the top  $N$  similarity weights and the corresponding sets of top preferences. In other words, for the top  $N$  neighbors of the active user, it outputs each user's preferences that are related to the current query and are not common with the active user's personal preferences selected, and these are candidate collaborative preferences.

**Example.** Assume that the algorithm selects the preferences related to the query from five user profiles that are shown in Figure 4. For simplicity, assume that each set of preferences derived contains five preferences. For these sets, the corresponding sets for common preferences are:

$$\begin{aligned} P'_1 &= (\{0.72, 0.9\}, \{0.63, 0.85\}, \{0.56, 0.5\}) \\ P'_2 &= (\{0.72, 0.8\}, \{0.56, 0.56\}) \\ P'_3 &= (\{0.63, 0.92\}, \{0.595, 0.7\}, \{0.56, 0.89\}) \\ P'_4 &= (\{0.72, 0.9\}, \{0.63, 0.65\}, \{0.595, 0.85\}, \\ &\quad \{0.56, 0.56\}, \{0.56, 0.7\}) \\ P'_5 &= (\{0.72, 0.85\}, \{0.63, 0.8\}, \{0.595, 0.65\}) \end{aligned}$$

All preferences of the fourth user are common with the active user's ones. Therefore,  $P'_4$  and  $P_4$  are discarded, since they can

contribute no new collaborative preferences. Also,  $P'_2$  ( $P_2$ ) is also discarded for having fewer common preferences than required. Finally, the following ( $N = 2$ ) sets of candidate collaborative preferences along with their similarity weights are produced:

$$P_1 = (\{p_{13}, 0.8\}, \{p_{14}, 0.78\}), \quad w_1 = 0.8$$

$$P_5 = (\{p_{51}, 0.85\}, \{p_{54}, 0.7\}), \quad w_5 = 0.694$$

### 4.3 Collaborative preference selection

This phase derives the set  $P_c$  of collaborative preferences that will affect the user query  $Q$  from the top preferences of each of the selected neighbours of the active user. For each such preference  $p_j$ , a prediction for the degree of interest of the active user is computed as the weighted sum of the degrees of interest of the neighbours in this preference:

$$d_j = \bar{d} + \frac{\sum w_i (d_{ij} - \bar{d}_i)}{\sum w_i}$$

where  $\bar{d}$  is the average degree of interest of the preferences of the active user,  $d_{ij}$  is the degree of interest of a preference of a neighbour, and  $\bar{d}_i$  is the average degree of interest of the neighbour. Collaborative preferences are ordered in decreasing predicted degree of interest and the top  $K_c$  preferences are selected for affecting the original user query.

**Example.** The top 2 collaborative preferences are:

```
MOVIE.mid = GENRE.mid
and GENRE.genre = 'thriller'      0.693
MOVIE.year > 1990                 0.647
```

### 4.4 Personalized answer generation

The final phase of collaborative query personalization is responsible for the generation of results that satisfy at least  $L$  of the  $K$  personal preferences and results that satisfy at least  $L_c$  of the  $K_c$  collaborative preferences. This is achieved by executing two queries, one integrating the set of personal preferences and the other integrating the set of collaborative preferences. Each query is formulated as the union of a set of sub-queries, each one corresponding to one or more of the  $K$  ( $K_c$ ) preferences selected. Each sub-query is built by extending the initial query by an appropriate qualification involving the participating preferences, and it returns along with its tuple the degree of interest of the preference satisfied. The expected results are obtained by taking the union of the partial results, grouping by the projected attributes of the initial query, and excluding all groups containing less than  $L$  ( $L_c$ ) rows, and they are ranked based on the degree of interest. This approach is called  $M_Q$  [19].

**Example.** These queries integrate the user's preferences:

```
Q1: select M.title, 0.72 degree
     from MOVIE M, GENRE G
     where M.mid = G.mid and genre = 'comedy'

Q2: select M.title, 0.56 degree
     from MOVIE M, GENRE G
     where M.mid = G.mid and genre = 'adventure'

Q3: select M.title, 0.63 degree
     from MOVIE M, DIRECTOR DI, DIRECTED DD
     where M.mid = DD.mid and DD.did = DI.did
```

```
and DI.name = 'D. Lynch'

Q4: select M.title, 0.595 degree
     from MOVIE M, ACTOR A, CAST C
     where M.mid = C.mid and C.aid =A.aid
     and A.name = 'A. Hopkins'

Q5: select M.title, 0.56 degree
     from MOVIE M, ACTOR A, CAST C
     where M.mid = C.mid and C.aid =A.aid
     and A.name = 'N. Kidman'
```

Preferences with the same implicit join may be combined using disjunction into a single sub-query. The personalized query qualifying results that satisfy at least  $L = 2$  of the top  $K = 5$  user preferences is this (each  $Q_i$  is replaced by its text):

```
select title, DEGREE_OF_CONJUNCTION(d)
from Q1 union all Q2 union all Q3
     union all Q4 union all Q5

group by title
having count(*) >= 2
order by DEGREE_OF_CONJUNCTION(d)
```

A user-defined aggregate,  $DEGREE\_OF\_CONJUNCTION$ , needs to be implemented and stored in the system. This function calculates the degree of interest of each group, i.e. the degree of interest of a tuple satisfying a set of preferences. Then, tuples returned are ordered in decreasing degree of interest by specifying an `order by` clause. Also, a personalized query describing results satisfying at least  $L_c = 1$  of top  $K_c = 2$  collaborative preferences is built.

## 5. EXPERIMENTAL EVALUATION

### 5.1 Experimental setup

We have implemented the algorithms described above on top of Oracle 9i and we have conducted experiments to compare their efficiency and effectiveness. Our data comes from the Internet Movies Database ([www.imdb.com](http://www.imdb.com)) and contains information about over 340000 movies. User profiles are stored in a separate table. For the experiments we have used synthetic user profiles and random queries.

We studied execution times for each phase of collaborative query personalization: (a) preference selection,  $PPS$  (b) neighbour identification,  $NEI$  (c) collaborative preference selection,  $CPS$  and (d) personalized answer generation,  $M_Q$ . The following parameters have been taken into account:

- $K$ : the number of top preferences from the user's profile that are related to the query and should affect the answer
- $L$ : the minimum number of those preferences that should at least be satisfied
- $K_c$ : the number of top collaborative preferences that are related to the query and should affect the answer
- $L_c$ : the minimum number of the collaborative preferences that should at least be satisfied
- $N_U$ : the number of users compared with the active one;
- $N$ : the number of neighbours selected

Each result shown below represents the average of 500 different experiment runs (50 profiles  $\times$  10 queries) with the same characteristics. All times are in seconds.

### 5.2 Experimental results

Figure 5 presents execution times for varying  $K$ . We have

considered that  $\kappa$  is also the number of related preferences of the active user for a query. For NEI, we have considered  $N_U = 30$ , and  $N = 4$ . For MQ, we have set parameters  $L = 1$ ,  $K_C = 1$ , and  $L_C = 1$ . We observe that execution times of CPS and PPS are negligible compared to the execution times of MQ and especially NEI. The latter is the most expensive algorithm of the whole procedure with respect to parameter  $\kappa$ . An interesting observation is the implicit dependence of NEI on the number of preferences  $\kappa$  selected from the active user's profile: as  $\kappa$  grows, there is an increased probability that more common preferences among the active user and other users will exist. In addition, there is also an increased probability that more users with common preferences with the active one will be found, thus, a larger number of them will be compared to the active one.

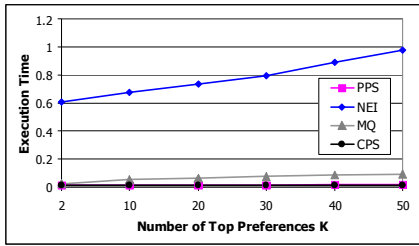


Figure 5. Execution times vs.  $K$

Figure 6 and 7 show execution times of MQ for different values of  $L$ , and  $K = 40$ ,  $K_C = 1$ ,  $L_C = 1$  and for different values of  $K_C$ , and  $K = 30$ ,  $L = 1$ ,  $L_C = 1$ , respectively. We observe that despite the fact that  $L$  is an input parameter for MQ, it does not affect the execution times of MQ. Similar results are taken for the execution times of MQ with respect to  $L_C$ .

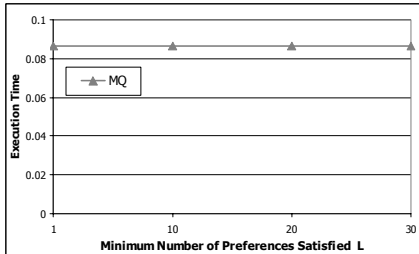


Figure 6. Execution times vs.  $L$

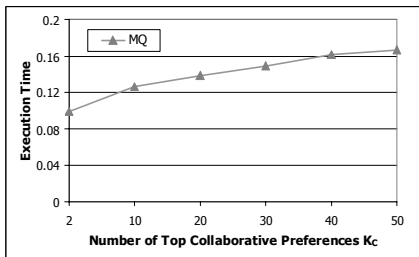


Figure 7. Execution times vs.  $K_C$

Figure 8 presents execution times of NEI for varying  $N_U$ . We have considered  $\kappa = 10$ ,  $N = 4$ . We observe that execution times depend heavily on the parameter  $N_U$ . Firstly, it selects the top

preferences of all these users, so the preference selection algorithm PPS has to be applied as many times as the number of users, i.e.  $N_U$  times. This results in a substantial increase in the overall execution time of NEI. Then, it spends a lot of time to find the common preferences each of the  $N_U$  users has with the active one. Moreover, there is also an implicit dependence of NEI on  $N_U$ : the more users considered the more users with common preferences with the active one may be found and the larger the overall number of common preferences to be processed may be.

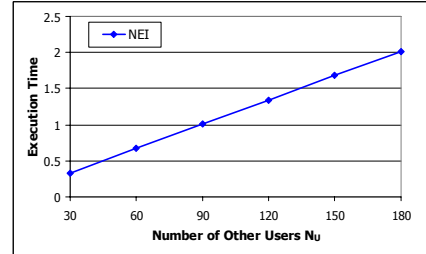


Figure 8. Execution times vs.  $N_U$

Overall, the most time-consuming phase of collaborative query personalization is neighbour identification followed by personalized answer generation. Execution times for these phases increase with  $\kappa$ ,  $K_C$ , and  $N_U$ . However, studies with users have shown that preferred values for  $\kappa$  and  $K_C$  are more likely to be up to 10. In this way, answers matching one's personal profile as well as answers matching collaborative preferences are smaller and more focused. In this case, the main issue remains execution time of the neighbour identification phase as a function of the number  $N_U$  of users that are compared with the active searcher. One direction is to apply clustering techniques off-line. In this way, the active user is compared with the users in the same cluster instead of all users in the system. However, when applying pre-clustering techniques, one has to bear in mind that there is a trade-off between accuracy and throughput. We are currently working on other techniques for the optimization of this phase, as well as the other phases of the collaborative query personalization system.

## 6. CONCLUSIONS AND FUTURE WORK

In this work, we have generalized earlier concepts of query personalization by introducing collaborative features, and we have defined collaborative query personalization, i.e. the process of dynamically enhancing user queries by taking into account personal as well as collaborative preferences. We have described a prototype system for collaborative query personalization over databases including results of experimental evaluation.

In ongoing work, we are planning experiments with users to solidify the evidence on the effectiveness of the approach and calibrate the parameters determining the effect of collaborative query personalization. Another open research area is automatic learning of these parameters that affect collaborative query personalization on a per user basis. Then, these could be also stored at the user profile.

We are also concerned with scalability and optimization issues of collaborative query personalization. Another challenging research direction is the support of collaborative query personalization for keyword queries.

Finally, since collaborative query personalization alters the search experience, the user interface needs to provide a way to explain what the system is doing to personalize the experience as well as to undo the personalization. Therefore, an interesting research direction is towards design of user interfaces that allow users to control the extent of the personalization, and can help alleviate inaccurate personalization.

## REFERENCES

- [1] Balabanovic, M., Shoham, Y. (1997). FAB: Content-Based, Collaborative Recommendation. *Comm. of ACM*, 40(3), 66-72, Mar. 1997
- [2] Birukov, A., Blanzieri, E., Giorgini, P. (2005). Implicit: An Agent-Based Recommendation System for Web Search. *AAMAS 2005*, 618-624
- [3] Billsus, D. Pazzani, M. (1998) Learning Collaborative Information Filters. *Proc. of ICML*
- [4] Briggs, P., Smyth, B. (2004). On the Use of Collaborative Filtering Techniques for the Prediction of Web Search Result Rank. *AH 2004, LNCS 3137*, pp. 380-383, 2004.
- [5] Bueno, D. David, A. (2001). METIORE: A Personalized Information Retrieval System. *UM01, LNAI 2109*, 168-177
- [6] Callan, J. (1996). Document Filtering with Inference Networks. *Proc. Of the ACM SIGIR Conf.*, 262-269
- [7] Chidlovskii, B., Glance, N. S., Grasso, M. A. (2000). Collaborative Re-Ranking of Search Results. *AAAI-2000 Workshop on AI for Web Search*
- [8] [www.eurekster.com](http://www.eurekster.com)
- [9] Fitzpatrick, L., Dent, M. (1997). Automatic feedback using past queries: Social searching? In *Proceedings of the 20th International ACM SIGIR, Philadelphia, PA, July 1997*.
- [10] Foltz, P., and Dumais, S. (1992). Personalized Information Delivery: An Analysis of Information Filtering Methods. *Comm. of the ACM*, 35(12), 51-60
- [11] Glance, N. S. (2000). Community search assistant. In *Artificial Intelligence for Web Search*, pp. 29-34, 2000.
- [12] Glover, E. J., Lawrence, S., Birmingham, W. P., Giles, C. (1999): Architecture of a Metasearch Engine that Supports User Information Needs. In *Proc. of CIKM 99*.
- [13] Herlocker, J., Konstan, J. A., Borchers, A., Riedl, J. (1999). An Algorithmic Framework for Performing Collaborative Filtering. In *Proc. of ACM SIGIR Conf.*, 230-237
- [14] [www.jeteye.com](http://www.jeteye.com)
- [15] Jeh, G. Widom, J. (2003). Scaling Personalized Web Search. In *Proc. of the 12th Int'l WWW2003*.
- [16] [www.jookster.com/JooksterPortal/resutls.aspx](http://www.jookster.com/JooksterPortal/resutls.aspx)
- [17] Koutrika, G., Ioannidis, Y. (2005). Constrained Optimalities in Query Personalization. In *Proc. of ACM SIGMOD*, 73-84, 13-16 June 2005.
- [18] Koutrika, G., Ioannidis, Y. (2005). Personalized Queries under a Generalized Preference Model. In *Proc. of 21st ICDE*, 841-852, 5-8 April 2005.
- [19] Koutrika, G., Ioannidis, Y. (2004). Personalization of Queries in Database systems. In *Proc. of 20th ICDE*, 597-608, 30 March - 2 April 2004.
- [20] Liu F., Yu C., Meng W. (2002). Personalized Web Search by Mapping User Queries to Categories. In *Proc. of CIKM'02*.
- [21] Loeb, S. (1992). Architecturing Personalized Delivery of Multimedia Information. *Comm. Of the ACM*, 35(12), 39-48, Dec. 1992
- [22] [myweb2.search.yahoo.com](http://myweb2.search.yahoo.com)
- [23] L. Page, S. Brin, R. Motwani, and T. Winograd (1998). The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford University Database Group.
- [24] Pitkow, J., Schutze, H., et al. (2002). Personalized Search. *Comm. of the ACM*, 45(9), 2002
- [25] Pujol, J., Sangüesa, R., Bermúdez, J. (2003). Porqpine: A Distributed and Collaborative Search Engine. *WWW2003*
- [26] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *Proc. of the ACM Conf. on Computer Supported Cooperative Work*, 75-186
- [27] Resnick, P. Varian, H. R. (1997). Recommender systems. *Comm. of the ACM*, 40(3), 56-58, Mar 1997
- [28] Shardanand, U., Maes, P. (1995). Social Information Filtering: Algorithms for Automating Word of Mouth. In *Proc. of ACM CHI'95 Conf.*, 210-217
- [29] Shahabi, C., Kashani, K. B., Chen, Y. S., McLeod, D. (2001). Yoda: An Accurate and Scalable Web-Based Recommendation System. *CoopIS'01, LNCS 2172*, 418-432
- [30] Sieg, A., Mobasher, B., Lytinen, S., Burke, R. (2003). Concept Based Query Enhancement in the ARCH Search Agent. *Int'l Conference on Internet Computing*: 613-619
- [31] Sullivan, D., Smyth, B., Wilson, D. (2005). Understanding Case Based Recommendation: A Similarity Knowledge Perspective. *Int'l Journal on Artificial Intelligence Tools*, Vol. 14, Nos. 1-2 (2005) 215-232
- [32] Tanudjaja, F., Mui, L. (2002). Persona: A Contextualized and Personalized Web Search. In *Proc. of the 35th Hawaii Int'l Conf. on System Sciences*
- [33] Teevan, J., Dumais, S., Horvitz, E.. Beyond the Commons: Investigating the Value of Personalizing Web Search. P. Brusilovsky, C. Callaway, A. Nürnberger (Eds.), *Proc. of the Workshop on New Technologies for Personalized Information Access (PIA 2005)* in conjunction with the 10th Int. Conf. on User Modeling (UM'05).
- [34] Yan, T., Garcia-Molina, H. (1995). SIFT: A Tool for Wide Area Information Dissemination. In *Proc. of USENIX Technical Conference*, 177-186.
- [35] Zaiane, O., Strilets, A. (2002). Finding Similar Queries to Satisfy Searches based on Query Traces. *EWIS 2002*