PersDB 2008

2nd International Workshop on
Personalized Access, Profile Management, and
Context Awareness: *Databases*

ELECTRONIC PROCEEDINGS

Sponsored by

23rd August 2008,
Auckland, New Zealand

(in conjunction with the 34th VLDB Conference)

Foreword

Proliferation of database-driven web sites has brought upon a plethora of applications where different notions of user and context information are of paramount importance. Monitoring and trading stock portfolios, blog aggregation and news notification, dataset sharing, weather tracking, and even simple search are just a few examples of applications that can all be personalized based on users' individual or social profiles and can be affected by their operational context, which may include the location, time, and other features of their environment. The trend towards more user-centric, personalized, and context-aware database systems requires new models and techniques able to provide users with the 'right information' at the 'right time' in the 'right place' and may affect database system functionality at several levels.

The PersDB 2008 workshop (http://persdb08.stanford.edu/) aimed at providing a forum for presentation of the latest research results, new technology developments, and new applications in the areas of personalized access, profile management, and context awareness in database systems. PersDB 2008 is the successor of the PersDL 2007 workshop (http://www.dblab.ece.ntua.gr/persdl2007/), and was kindly sponsored by the DELOS Association for Digital Libraries (http://www.delos.info/index.php?option=com_content&task=view&id=614&Itemid =342).

The workshop was held in Auckland, N. Zealand, on August 23, 2008, in conjunction with the VLDB 2008 Conference. It included 1 keynote talk, 7 paper presentations, and a panel. All papers as well as an extended abstract of the keynote talks are included here.

Letizia Tanca (Politecnico di Milano) gave the keynote talk. Letizia is an expert in context-aware database design, data management for mobile and pervasive systems, and dynamic, semantics-based database integration. In her talk, she analysed the most interesting approaches to context modeling and usage and particularly focused on a context model and a methodology appropriate for data tailoring.

The program committee for this workshop consisted of 22 members and was chaired by Vassilis Christophides (IST-FORTH, Hellas) and Georgia Koutrika (Stanford University, USA). It accepted 7 papers out of 11 submissions.

We would like to thank all the people who have supported and helped in the organization of this workshop: the authors and presenters of the papers, the reviewers for their effort and help in preparing the workshop's program, and the organizers. We would also like to thank Yannis Stavrakas (Institute for the Management of Information Systems, Hellas) for maintaining the workshop's web site and preparing the e-proceedings.

Program Committee Chairs
Vassilis Christophides (IST-FORTH, Hellas)
Georgia Koutrika (Stanford University, USA)

General Chairs
Tiziana Catarci (Università di Roma "La Sapienza", Italy)
Yannis Ioannidis (University of Athens, Hellas)
Timos Sellis (Institute for the Management of Information Systems, Hellas)

Workshop Officers

Steering Committee

- Tiziana Catarci (Universita di Roma "La Sapienza", Italy), catarci@dis.uniroma1.it
- Yannis Ioannidis (University of Athens, Greece), yannis@di.uoa.gr
- Timos Sellis (Institute for the Management of Information Systems and National Technical University of Athens, Greece), timos@imis.athena-innovation.gr

Program Committee Chairs

- Vassilis Christophides (ICS-FORTH, Heraklion, Greece), christop@ics.forth.gr
- Georgia Koutrika (Stanford University, USA), koutrika@stanford.edu

Program Committee

- Grigoris Antoniou (ICS-FORTH, Greece)
- Ricardo Baeza-Yates (Yahoo! Research, Spain and Chile)
- Wolf Tilo Balke (University of Hannover, Germany)
- Jan Chomicki (University at Buffalo, USA)
- Paolo Ciaccia (University of Bologna, Italy)
- Ling Feng (Tsinghua University, China)
- Irene Fundulaki (ICS-FORTH, Greece)
- Werner Kiessling (University of Ausburg, Germany)
- Masaru Kitsuregawa (University of Tokyo, Japan)
- Nikos Koudas (University of Toronto, Canada)
- Alexandros Labrinidis (University of Pittsburgh, USA)
- Carlo Meghini (CNR, Italy)
- Massimo Melucci (University of Padua, Italy)
- Bamshad Mobasher (DePaul University, USA)
- Wolfgang Nejdl (University of Hannover, Germany)
- Moira Norrie (ETH-Zentrum, Switzerland)
- Christos Papatheodorou (Ionian University, Greece)
- Evi Pitoura (University of Ioannina, Greece)
- Guillaume Raschia (Polytech'Nantes, France)
- Nicolas Spyratos (University of Paris-South, France)
- Yannis Stavrakas (Institute for the Management of Information Systems, Greece)
- Martin Theobald (Stanford University, USA)
- Panayiotis Tsaparas (Microsoft Research, USA)
- Xiaofang Zhou (The University of Queensland, Australia)

Personalizing the Search for Knowledge

Minko Dudev
Max-Planck Institute for Informatics
Saarbrücken, Germany
mdudev@mpi-inf.mpg.de

Shady Elbassuoni
Max-Planck Institute for Informatics
Saarbrücken, Germany
elbass@mpi-inf.mpg.de

Julia Luxenburger
Max-Planck Institute for Informatics
Saarbrücken, Germany
julialux@mpi-inf.mpg.de

Maya Ramanath
Max-Planck Institute for Informatics
Saarbrücken, Germany
ramanath@mpi-inf.mpg.de

Gerhard Weikum
Max-Planck Institute for Informatics
Saarbrücken, Germany
weikum@mpi-inf.mpg.de

ABSTRACT

Recent work on building semantic search engines has given rise to large graph-based knowledge repositories and facilities for querying them and more importantly, ranking the results. While the ranking provided may prove to be acceptable in general, for a truly satisfactory search experience, it is necessary to tailor the results according to the user's interest. In this paper, we address the issue of personalizing query results in the specific setting of graph-based knowledge bases. In particular, we address two important issues: i) construction of the user profile based on the inference of the user's interest and ii) a formal model for personalized scoring which incorporates the user's interest. Preliminary experimental results show that our techniques are indeed promising.

1. INTRODUCTION

Personalization of search has been named as one of the next big challenges in information retrieval. Understanding the user and the context of her search is crucial in satisfying her information need and drastically reducing the time needed to find the "right" information. Personalization research has encompassed a wide variety of tasks, including, analyzing user click-stream data, generating user profiles, result re-ranking techniques based on user profiles, architectures for personalization, etc. [15, 13]. The main focus of this research is on the personalization of search results in the context of documents and keyword-based web search.

However, while the web is the largest repository of data, it is also, for the most part, unstructured. Several recent efforts have gone towards the building of large "knowledge" repositories – knowledge bases containing structured data extracted from the web in the form of entities and relationships (for example, Freebase[1], YAGO [14], etc.). Systems such as NAGA [7] and ExDBMS [3] provide query processing facilities on such repositories. Semantic web standards such as RDF and SPARQL aim at supporting a consistent representation and querying of such semantic, structured data. With the growth of structured knowledge bases, search result ranking and personalization again become key issues in satisfying a user's information need.

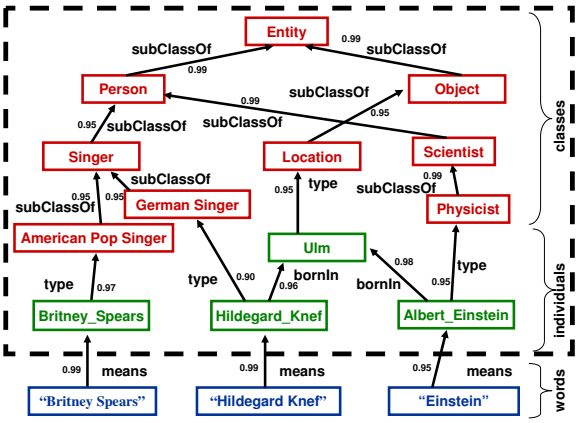

Figure 1: Portion of the YAGO knowledge graph.

In this paper, we are concerned with personalizing query results on graph-based knowledge bases. In particular, we develop techniques for personalization in the context of NAGA [7], a semantic search engine. In addition to providing query processing over a large knowledge base, NAGA also provides a scoring model which can be used as a basis to test our techniques. NAGA's knowledge base consists of several millions of facts (see YAGO [14] for details), provides a graph-based query language and returns graphs as results. A snippet of the knowledge base is shown in Figure 1. NAGA's data model is a graph, in which the nodes represent entities and the edges represent relationships between the entities. An edge in the graph with its two end-nodes forms a *fact*. As facts have been automatically extracted from the Web each one is associated with a confidence value to reflect the trust in the extraction process. It is possible to ask queries such as "When was Einstein born?" (<Einstein BORNIN $x>), "What do Britney Spears and Hildegard Knef have in com-

[1] http://www.freebase.com

2nd International Workshop on Personalized Access, Profile Management, and Context Awareness: Databases (*PersDB'08*), August 23, 2008, Auckland, New Zealand

mon?" (<Spears CONNECT Knef> – they are both singers), etc. The results returned by NAGA are ranked according to a scoring model which takes into account the confidence of facts in the results (higher confidence facts are preferred), informativeness ("popular" facts preferred) and compactness ("tightly connected" answers preferred).

NAGA's scoring model has been shown previously to be useful and effective (see [7] for details). Our aim is to further enhance its effectiveness by incorporating user interest into the scoring model. As a concrete example, suppose a user issues the query <$x ISA singer> in order to get a list of singers. The original NAGA scoring model ranks "Britney Spears" at the top. This would be acceptable if there were no prior information about the kind of music the user prefers. However, if it was known that the user had shown an interest in German music (possibly through a previous query or browsing session), then ranking "Hildegard Knef" and other German singers at the top might be of more interest to her.

Our work addresses two specific problems encountered in the personalization of structured search:

First, we propose methods to infer a user's interest in various topics based on her interest in a limited number of facts and entities. We regard the user profile as a snippet of the knowledge graph and utilize the ontological facts present in the knowledge base to propagate scores from user-accessed entities and facts.

Second, we present a formal model for personalization which can be easily incorporated into any probabilistic scoring model. Our model is based on generative language models and incorporates personalized entity and relationship scores into the ranking. In particular, we show how to incorporate our personalized scoring model into NAGA's ranking function.

Some previous work exists on the personalization of query results on structured data. For example, [8] develops techniques for personalizing results on relational databases and Piment [1] does the same for XML queries. We differ considerably from these works in both the setting (schemaless, graph-structured data) and approach (use of ontological facts to infer preferences). Techniques to use ontologies to derive profiles have been described in [6, 13]. While we too use ontological facts to derive user profiles, our methods for doing so and our setting (graph-based knowledge base as opposed to documents) are different.

Contributions and Outline. In a nutshell, we make the following contributions.

- a conceptualization of user profiles as snippets of the knowledge graph and methods to infer user interest across this graph.
- a probabilistic personalized ranking model, and
- a preliminary experimental evaluation as a proof-of-concept.

The remainder of the paper is organized as follows. In Section 2 we provide some background information on NAGA, the semantic search engine over which we have implemented and tested our techniques. Section 3 describes our method to generate the user profile. We then describe how to make use of the constructed user profile in order to perform personalization in Section 4. In Section 5 we present a preliminary evaluation to assess the effectiveness of our approach. We discuss related work in Section 6 and conclude in Section 7.

2. BACKGROUND

Our work is motivated by large graph-based knowledge repositories such as [14]. While our techniques can be adapted and applied to any system where the underlying knowledge-base is a graph and the query results are graphs or trees, in this work, we incorporate our personalization strategies in the framework provided by NAGA [7], a new semantic search engine. In this section we provide a brief overview of NAGA and outline details of its ranking function.

NAGA is a semantic search engine with a large knowledge-graph of facts. It's query language is based on SPARQL and the results are naturally regarded as graphs. As mentioned in the introduction, NAGA's scoring model scores answer graphs based on confidence (higher confidence facts are preferred), informativeness ("popular" facts are preferred) and compactness ("tightly connected" answers are preferred), which are integrated into a unified framework. Its approach is inspired by existing work on language models for information retrieval on document collections, and is adapted and extended to the new domain of labeled and weighted graphs. Here we outline how the scoring model is derived and refer the reader to [7] for additional details.

NAGA's scoring model assumes that a query q is generated by a probabilistic model of a result graph g. A query q is represented as $q = q_1 q_2 ... q_k$, where q_i is a *fact template* (that is, if a fact is represented as <x R y>, a fact template has at least one of x, R or y unbound). Analogously, a result graph g is represented as $g = g_1 g_2 ... g_k$, where g_i is a fact matching the fact template q_i. The ranking of a result graph is based on $P(g|q)$, which is the probability that the result graph g generated the (observed) query q. After applying Bayes formula and dropping a graph-independent constant, we have: $P(g|q) \sim P(q|g)P(g)$ where $P(g)$ is the prior and is assumed to be uniform. We now estimate $P(q|g)$ as: $P(q|g) = \prod_{i=1}^{n} P(q_i|g)$. The likelihood of a fact template given an answer graph is now modeled as a mixture of two distributions, $\tilde{P}(q_i|g)$ and $\tilde{P}(q_i)$ as follows:

$$P(q_i|g) = \alpha \cdot \tilde{P}(q_i|g) + (1-\alpha) \cdot \tilde{P}(q_i), 0 \leq \alpha \leq 1 \qquad (1)$$

$\tilde{P}(q_i|g)$ is the probability of drawing q_i randomly from an answer graph, $\tilde{P}(q_i)$ is the probability of drawing q_i randomly from the total knowledge graph and α is either automatically learned (via EM iterations [4]) or set to an empirically calibrated global value. [4, 17] show the connection between this style of probabilistic models and the popular $tf \cdot idf$ heuristics.

In order to capture confidence, informativeness and compactness, $\tilde{P}(q_i|g)$ is modeled by a mixture model which puts different weights on confidence and informativeness. This is close in spirit to linear interpolation models used for smoothing [17].

$$\tilde{P}(q_i|g) = \beta \cdot P_{conf}(q_i|g) + (1-\beta) \cdot P_{info}(q_i|g) \qquad (2)$$
$$0 \leq \beta \leq 1$$

For details of the estimation of each of the component probabilities as well as the background model $\tilde{P}(q_i)$, we refer the reader to [7]. In summary, we use P_{NAGA} to refer

to the ranking provided by NAGA. We develop a new scoring model P_{user} which ranks results based only on the user preference. We then combine P_{user} with P_{NAGA} to get the personalized scoring model, $P_{personalized}$. In the next section, we describe the generation of user profiles and then develop the model for $P_{personalized}$ in Section 4.

3. USER PROFILE

A user profile corresponding to a single user is, conceptually, the knowledge graph with interest scores attached to entities and facts. Intuitively, we associate interest scores to entities and facts which have been accessed by the user[2]. However, since the number of elements which the user accesses is a very small fraction of the actual knowledge base, we allow propagation of interest scores to the neighbors of the accessed entities, as well as to related facts in order to reason about the context of her interests and to incorporate them in future queries. We discuss how these scores are assigned and propagated in this section.

3.1 Entity Score Assignment and Propagation

We first consider the problem of assigning and propagating interest scores to entities.

Score Assignment. Initially, all entities have an interest score of $\epsilon > 0$, a very small default interest score. When the user accesses an entity k, the interest score of that entity $P_{in}(k)$ is updated to:

$$P_{in}(k) = \frac{\#accesses(k)}{\Sigma_n \#accesses(n)}$$

where the numerator is the total number of times k has been accessed and the denominator is the total number of entity accesses in the knowledge graph.

Score Propagation. The interest score described only accounts for the entities the user already knows. A personalization based only on these interest scores would limit the user in her desire of discovering new entities, and would miss the opportunity of generalizing more abstract user interests from the observed user access patterns. And so, in order to infer the user's interest, we allow for the propagation of scores from accessed entities.

Given that an entity k was accessed, we first determine which other entities are candidates for score propagation. A natural candidate is the class of the entity accessed. Consider the example depicted in Figure 2. In the first iteration, the entity Britney_Spears is accessed. Thus, we infer in the second iteration that the user may be interested in other entities of the class American_Pop_Singer. And so, we propagate a portion of the updated interest score for Britney_Spears to the American_Pop_Singer. Similarly, we can also propagate a portion of the score to other entities of the class American_Pop_Singer (for example, Sheryl_Crow). To avoid cycles, each edge can be traversed at most once during propagation. Thus there is no interest score propagated back from American_Pop_Singer to Britney_Spears in this step. Further, we continue to propagate interest scores up

[2]We assume that we are able to determine user interest in facts and entities when she accesses them through an appropriate user interface.

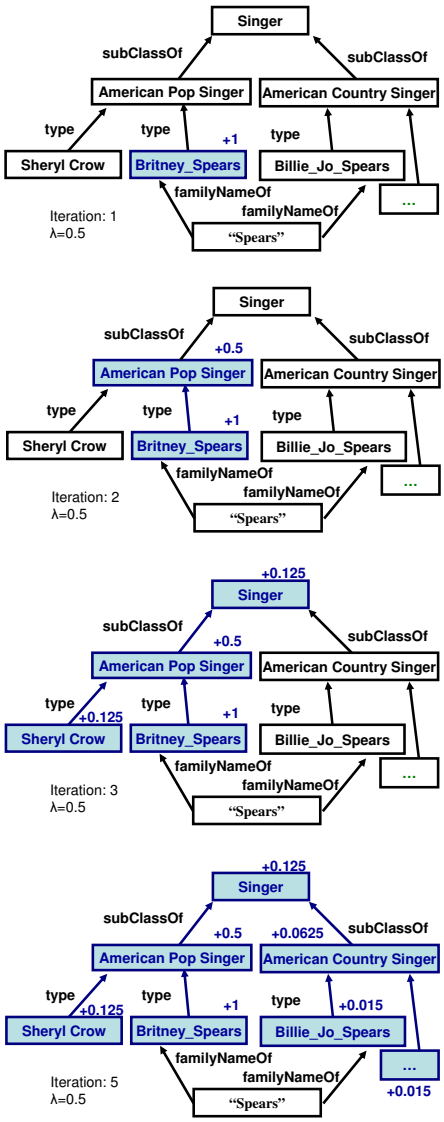

Figure 2: Entity score propagation

the class hierarchy until the root node entity. And so, the interest scores can flow from entity to class, from class to class and from class to entity, but not from entity to entity. Thus in our sample propagation, there is no flow of interest from Britney_Spears to Spears across the FAMILYNAMEOF relationship. This is to prevent interest scores from topic drifts to rather unrelated concepts. E.g., via the relationship FAMILYNAMEOF Britney_Spears is connected to Heather_Spears, a Canadian artist and poet.

More formally, our entity interest propagation scheme can be described as follows. Let k be the node whose interest score was updated. Let Q be the set of its qualifying neighbors which are eligible for score propagation (that is, nodes reachable through the edges labeled SUBCLASSOF and TYPE), but to which the score has not yet been propagated. For each $i \in Q$, we update the interest score using the formula:

$$P^*_{in}(i) = P_{in}(i) + \frac{P_{in}(k) \cdot \lambda}{|Q|}$$

where the left hand side denotes the updated interest score for entity i, $0 \leq \lambda \leq 1$ is a *damping factor* to control the amount of score to be propagated, and $|Q|$ is the number of qualifying nodes. The above formula propagates equal damped scores to each qualifying node. The propagation is repeated for each entity in Q until there are no more entities left, or the propagated score mass $(P_{in}(i) \cdot \lambda)$ is less than a pre-defined threshold. Finally, a normalization of the obtained entity interest scores lets us cast $P^*_{in}(i)$ into the probability of user interest in entity i.

3.2 Fact Score Assignment and Propagation

The user may be interested in two different entities in the knowledge graph. If these entities are connected by a relationship, we may be tempted to infer that the user is interested in the fact denoted by these two nodes and the edge connecting them. However, since the user could have independently accessed either entity in different contexts, this inference may not be valid. And so, we assign interest scores to facts separately.

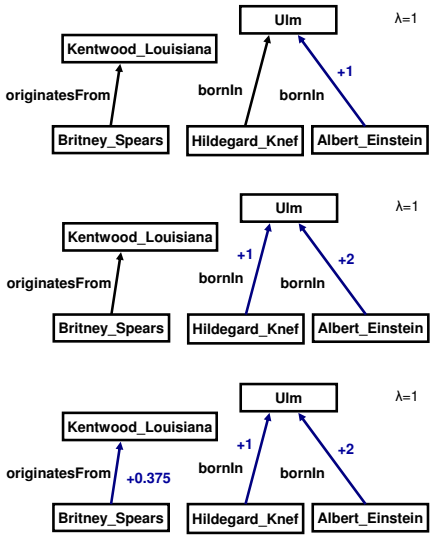

Figure 3: Fact score propagation

Score Assignment. Initially, all facts have an interest score of $\epsilon > 0$, a very small default score. Analogous to the case of entities, when the user accesses a fact f, the interest score of that fact is updated to:

$$P_{in}(f) = \frac{\#accesses(f)}{\Sigma_n \#accesses(n)}$$

where the numerator is the total number of times f has been accessed and the denominator is the total number of fact accesses in the knowledge graph.

Score Propagation. In order to infer the user's interests, we need to propagate interest scores from specific facts to other "related" facts. We need to first identify candidates for score propagation. We have at least a couple of cases to consider here: i) facts with the same relation as the accessed fact, ii) facts with "similar" relationship types as the accessed facts.

For the first case, we propagate equal score to each fact with the same relation as the accessed fact. Let f be the edge accessed. Let F be the set of facts with the same relationship as f. For each fact $i \in F$, we update its score to:

$$P^*_{in}(i) = P_{in}(i) + \frac{P_{in}(f) \cdot \lambda}{|F|}$$

where the left hand side denotes the updated interest score for fact i, $0 \leq \lambda \leq 1$ is a damping factor to control the amount of score to be propagated. As with the case of entities, only a damped score is added to each of the other facts. In the example in Figure 3, interest is thus propagated from the fact <Albert_Einstein BORNIN Ulm> to <Hildegard_Knef BORNIN Ulm>.

Next, we consider relationships which are similar to the relation of the accessed fact. For example, BORNIN and ORIGINATESFROM are similar relationships since the entities that can have these relationships have the same classes – `person` and `location`. However, the same entity may belong to different classes. And so, in order to more precisely quantify the similarity, we utilize a similarity metric. Let <x R y> denote a fact. Let $left$ denote the set of classes to which x belongs and let $right$ denote the set of classes to which y belongs. Note that all entities belong to the class entity. Let i and j be two sets of facts corresponding to relations R_i and R_j respectively. The similarity between R_i and R_j is computed as follows:

$$similarity_{ij} = \left(\frac{|left_i \cap left_j|}{max(|left_i|, |left_j|)} + \frac{|right_i \cap right_j|}{max(|right_i|, |right_j|)}\right)/2$$

Let f be the edge accessed and R be the set of relationships in the knowledge graph which are different from the relation of f. For each $i \in R$, we first compute the similarity score $S(i)$ using the formula above. Let F_i be the set of facts with relationship i. For each fact $f_i \in F_i$, we update the interest score as follows:

$$P^*_{in}(f_i) = P_{in}(f_i) + \frac{P_{in}(f) \cdot \lambda}{|F_i|} \cdot S(i)$$

where λ is the damping factor. Note that the similarity $S(i)$ acts as a weighting factor to give higher weight to facts with highly similar relations as compared to facts with less similar relations. In our example in Figure 3, this means that interest is propagated from <Albert_Einstein BORNIN Ulm> to the fact <Britney_Spears ORIGINATESFROM Kentwood _Louisiana> as follows. Each of the entities involved in the two facts belong to the following classes.

Albert_Einstein	$\in$	Physicists, German_Americans, Pacifists, person
Britney_Spears	$\in$	Singers, English_Americans, Actors, person
Ulm	$\in$	Cities_in_Baden_Württemberg, city, municipality, location
Kentwood_Louisiana	$\in$	Towns_in_Louisiana, town, municipality, location

Then the similarity between the two facts amounts to $(0.25 + 0.5)/2 = 0.375$. And so, a score of 0.375 is propagated from the first fact with Einstein to the second fact with Spears. Again, a final normalization step allows us to treat interest scores $P^*_{in}(f)$ as the probability of user interest in the fact f.

4. PERSONALIZATION STRATEGY

We cast our personalized ranking approach in a probabilistic model. To allow for a tuning of the strength of the employed personalization at any time, we define our personalized ranking function as a mixture of the original ranking function $P_{NAGA}(g|q)$ of NAGA and the user-biased ranking function $P_{user}(g|q)$ as follows:

$$P_{personalized}(g|q) = \gamma.P_{NAGA}(g|q) + (1-\gamma).P_{user}(g|q)$$

Thus by choosing γ appropriately, the influence of the user-specific ranking bias can be regulated.

In the following, we derive our proposal for estimating the user-specific probability that the answer graph g is relevant to the query q. Our derivation of a personalized ranking function follows similar lines as the derivation of the original NAGA ranking, thus transferring the reasoning from the complete knowledge graph to the user-specific graph snippet. We also adopt the notion of a background model weighting the fact templates comprising the query against each other, as well as, informativeness which measures the popularity of facts matching the query. While informativeness based on the complete knowledge graph would, e.g., give high weight to the entity "Britney Spears", informativeness on the personal knowledge graph snippet might deem "Hildegard Knef" more popular, and thus more interesting to the user.

More formally, the user-specific ranking function $P_{user}(g|q)$ can be re-written as $P_{user}(g|q) \approx P_{user}(q|g) \cdot P_{user}(g)$ using Bayes' rule. The denominator $P_{user}(q)$ can be dropped as it is the same for all answer graphs and thus has no influence on the ranking order. With $P_{user}(g)$ assumed to be uniform, we are left with $P_{user}(q|g)$, the probability that the answer graph g generated the query q given the user profile.

Recall that a query is expressed as $q = q_1 q_2 ... q_k$ where q_i is a fact template. A fact template is of the form <x R y> where at least one of x, R or y is unbound. Assuming independence between fact templates, we have

$$P_{user}(q|g) = \prod_{i=1}^{n} P_{user}(q_i|g)$$

Again we apply Bayes' rule to obtain $P_{user}(q_i|g) \approx P_{user}(g|q_i) \cdot P_{user}(q_i)$. $P_{user}(g|q_i)$ is the probability that the answer graph g matches the fact template q_i. $P_{user}(g|q_i)$ assesses the informativeness of the answer graph given the query template and user profile, while $P_{user}(q_i)$ serves the purpose of a user-specific background model.

4.1 Definition of the user-specific background model

The background model weights the different fact templates in the query (this is similar in spirit to term weighting in standard IR). In order to weight the fact templates, we need to reason on the *bound* parts of each fact template q_i, and weight different query units against each other depending on the user interest. That is

$P_{user}(q_i) =$

$$\begin{cases} P^*_{in}(x) & \text{if only x is bound in } q_i \\ P^*_{in}(y) & \text{if only y is bound in } q_i \\ P^*_{in}(x) \cdot P^*_{in}(y) & \text{if x, y are bound in } q_i \\ \sum_{f'=(s,R,t)} P^*_{in}(f') & \text{if only R is bound in } q_i \\ P^*_{in}(x) \cdot \sum_{f'=(s,R,t)} P^*_{in}(f') & \text{if x, R are bound in } q_i \\ P^*_{in}(y) \cdot \sum_{f'=(s,R,t)} P^*_{in}(f') & \text{if R, y are bound in } q_i \end{cases}$$

Clearly, when only one of the entities is bound, the user interest boils down to her interest in that particular entity. Our score propagation scheme ensures that both direct as well as indirect interest in the entity play a role in the final interest score. However, when both entities are bound, we have choice of either multiplying or adding the user interest in those entities. We chose multiplication based on the intuition that interest in two entities may have been expressed independently of each other.

The last three cases deal with fact templates when the relation is bound. When only the relation is bound, the user interest in that relation is the sum of interest in all facts containing that relation. Note that if the user has previously expressed interest in very few facts containing this relation, our propagation takes care that other facts with the same relation receive only a damped score. As in the previous case, if an entity as well as a relation are both bound, we then assume independence in the user interest in each of them and multiply the interest scores.

4.2 Definition of the user-specific informativeness

Next we estimate the user-specific informativeness $P_{user}(g|q_i)$ which serves a similar purpose as informativeness in the original NAGA ranking. Instead of assessing the overall popularity of facts, it measures popularity with respect to the user-specific snippet of the knowledge graph. Since the bound arguments of a query q, comprising of fact templates, are the same for each answer graph g, the user-specific informativeness $P_{user}(g|q_i)$, estimates the user interest in the *unbound* parts of the query. Since each q_i has a match f, $P_{user}(g|q_i)$ reduces to $P_{user}(f|q_i)$.

The probability of a fact $f = <x'\ R'\ y'>$ matching $q_i =$ <x R y> in the context of the user profile, $P_{user}(f|q_i)$, is defined as the probability of user interest in the unbound parts of the fact f, given that we just learnt from the user formulating the query q that she is interested in the bound parts of f.

We estimate this probability by reasoning on the interest in the unbound parts of f, i.e.,

$P_{user}(f|q_i) =$

$$\begin{cases} P^*_{in}(x') & \text{if x unbound in } q_i \\ P^*_{in}(y') & \text{if y unbound in } q_i \\ P^*_{in}(x') \cdot P^*_{in}(y') & \text{if x, y unbound in } q_i \\ \sum_{f'=(s,R',t)} P^*_{in}(f') & \text{if R unbound in } q_i \\ P^*_{in}(x') \cdot \sum_{f'=(s,R',t)} P^*_{in}(f') & \text{if x, R unbound in } q_i \\ \sum_{f'=(s,R',t)} P^*_{in}(f') \cdot P^*_{in}(y') & \text{if R, y unbound in } q_i \\ P^*_{in}(x') \cdot \sum_{f'=(s,R',t)} P^*_{in}(f') \cdot P^*_{in}(y') & \text{else} \end{cases}$$

Here we assume independence, and thus consider the products of user interests in the unbound parts of f. The general user interest into the relation R is obtained by aggregating the user interest in facts containing a relation R.

Example. Suppose a user accesses the fact <Britney_Spears ISA singer>. Some time later, she poses the query $q = <\$x$ BORNIN Ulm> $\wedge$ <$x ISA $y> i.e., a query consisting of two templates $q_1 = <\$x$ BORNIN Ulm> and $q_2 = <\$x$ ISA $\$y>$. Then the user-specific background model, $P_{user}(q_i)$ weights the two templates q_1 and q_2 against each other based on the probability of user interest in the bound query parts, i.e., the user interest in BORNIN Ulm with respect to ISA. For each template the facts matching the template are ranked based on their unbound parts. E.g., let the two candidate answers be g_1 = <Albert_Einstein BORNIN Ulm> $\wedge$ <Albert_Einstein ISA physicist> and g_2 = <Hildegard_Knef BORNIN Ulm> $\wedge$ <Hildegard_Knef ISA singer> Now, g_1 and g_2 are ranked based on P_{user}(Albert_Einstein$|q_1$) and P_{user}(Albert_Einstein, physicist$|q_2$), respectively P_{user}(Hildegard_Knef$|q_1$) and P_{user}(Hildegard_Knef,singer$|q_2$). In the original NAGA ranking, we would expect the informativeness of Albert_Einstein and physicist to be higher than the informativeness of Hildegard_Knef and singer due to the popularity of Albert Einstein. In the context of the user profile which documented a user interest in singers, we would, however, expect Hildegard Knef to have the higher user-specific informativeness.

5. EXPERIMENTAL EVALUATION

5.1 Experimental Setup

We use the NAGA semantic search engine to test our score propagation and personalization techniques. At the time of this evaluation, we did not have access to user logs[3]. Instead, we created our own queries and profiles to specifically test the various components of our techniques.

Profile	Topic	Query used to build profile
1.	OpenGL	What is OpenGL?
2.	Java (the PL)	All facts about Java
3.	Kevin Mitnick	All facts about Kevin Mitnick
4.	Britney Spears	All facts about Britney Spears
5.	Friedrich Nietzsche	All books written by Nietzsche
6.	Yao Ming	All facts about Yao Ming
7.	Pirates of the Caribbean	All actors of the Pirates of the Caribbean movie
8.	Levenshtein distance	All facts about Levenshtein distance
9.	Steven Seagal	Movies in which Seagal acted
10.	Picasso	All of Picasso's creations

Table 1: User profiles generated from topics

Creation of user profiles. In order to create user profiles, we chose topics from the INEX 2007 Adhoc Track[4]. The track makes use of a subset of Wikipedia and around 450 topics are proposed. Each topic is usually based on an entity. We chose 10 of those topics which were also available in NAGA's knowledge base as the basis for generating user profiles. Table 1 lists these topics. For each topic, the query listed in the third column was fired and the top-10 results were taken as interesting to the user. The entities as well as facts corresponding to each of the results were deemed to be clicked by the user. These clicks were then translated into scores and propagated to other parts of the knowledge base. For example, for profile number 5, the query fired was: <Friedrich_Nietzche WROTE $x>. The top-10 results included entity bindings for $x (for example, "The birth of tragedy"). Each of these entities were assumed to be clicked. In addition, the fact corresponding to the result ("Friedrich_Nietzsche wrote The_birth_of_tragedy") was also assumed to be clicked.

Profile	Topic	Example Query	Expected Results
Re-finding Queries			
1.	OpenGL	$x is Graphics_library	OpenGL on top
2.	Java (the PL)	$x $r Sun_Microsystems	Java on top
3.	Kevin Mitnick	$x type Computer_Security_Specialist	Mitnick's name on top
4.	Britney Spears	$x isa American_dance_musician	Britney Spears on top
Queries testing entity interest propagation			
5.	Friedrich Nietzsche	$x isa book $x subClassOf book	Nietzsche's books, books on philosophy The class of Nietzsche's books, the category philosophical books on top
6.	Yao Ming	$x subClassOf Player	The classes Houston Rockets players, basketball players on top
7.	Pirates of the Caribbean	$x subClassOf movie	The classes Pirate films, action films on top
8.	Levenshtein distance	$x subClassOf algorithm	String algorithms on top
Queries testing fact interest propagation			
9.	Steven Seagal	$x subClassOf movie Woody_Allen $r $x	Action thrillers, American movies movies in which Allen was an *actor*
10.	Picasso	Vincent_van_Gogh $r $x	van Gogh's *paintings*

Table 2: Queries generated to test personalization

Queries. Once each of the above profiles was set up, we formulated queries to test our techniques. Table 2 lists some examples of queries we constructed and the results we expected to see. The queries were designed to specifically test the effects of the individual components of the user-specific informativeness. The first set of queries in Table 2 makes a case for remembering previous accesses to entities. For example, query 2 makes use of the clicks on the entity Java_PL (programming language) when the user searches for interesting facts about Sun Microsystems. The next set of queries we considered was conceived to test our entity score propagation. For example, on profile 5 the goal was to

[3]Note that the collection of user logs is work in progress.
[4]`inex.is.informatik.uni-duisburg.de/2007/`

test the entity score propagations from Nietzsche's books to other books (e.g., The_birth_of_tragedy → category philosophy books → other instances of philosophy books), as well as, from Nietzsche himself to other philosophers. Finally, we constructed queries to witness the effects of our fact score propagation scheme. Consider for example query 10. Since the profile was generated based on Picasso's creations as an artist, we expect the listing of van Gogh's paintings to be ranked high, rather than his personal details (such as, birth date, birth place, etc.). Similarly, profile 9 is concerned with the movies in which Seagal acted in and so, for query 9, we would expect a list of movies in which Woody Allen *acted* in, as opposed to his personal details or movies that he directed.

5.2 Results

Rank	NAGA	Personalization
1.	I, Robot	The Antichrist
2.	I, Libertine	The Birth of Tragedy
3.	I, Claudius	On the Genealogy of Morality
4.	Little, Big	The Gay Science
5.	Contact	Ecce Homo
6.	V.	Nietzsche contra Wagner
7.	Night	Twilight of the Idols
8.	Magic, Inc	In a Glass Darkly
9.	Sex	The Book on Adler
10.	Girl, Interrupted	The Autobiography of Charles Darwin

Table 3: Results for the query <$x isa book> based on the profile generated by <Friedrich_Nietzsche wrote $x>

On our test queries, our personalization approach coincides with our expectations. In the following we discuss a couple of the more interesting examples testing our propagation scheme in more detail. We first consider the queries on profile 5. Table 3 shows the results and ranking returned by both NAGA as well as our personalization[5] for the first query <$x ISA book>. As expected, the books by Nietzsche ranked very high – in fact, the first 7 results are books by Nietzsche. In addition, we also have a book written by Kierkegaard, another well-known philosopher, at rank 9 (The Book on Adler) showing that propagation and personalization are boosting the right results to the top. How the books reported at position 8 (In a Glass Darkly by Sheridan Le Fanu) and position 10 (The Autobiography of Charles Darwin by Charles Darwin) connect to Nietzsche is not as obvious since neither of them is a philosophical book nor written by a philosopher. A closer look at the propagation revealed that Nietzsche's book "The Birth of Tragedy" was written in 1872, the same year that the above two books were written. Since "The Birth of Tragedy" also belongs to the category "1872 books", the score was propagated to the other two books as well. Thus, our propagation inferred an interest in books written in the times of Nietzsche.

Similarly, our second query on this profile exhibits effects of our entity score propagation. Suppose a user had shown interest in Nietzsche's work and was now interested in browsing other books in the knowledge base, he could issue the query <$x SUBCLASSOF book>. The results for this query are reported in Table 4. Consider the personalized results.

[5]Note that the result ranking presented here are based only on the personalized ranking. We did not mix the original NAGA score and the personalized score since our goal was to test only the personalization techniques.

Rank	NAGA	Personalization
1.	formulary	Books_by_Nietzsche
2.	curiosa	Philosophy_books
3.	authority	1895_books
4.	booklet	1889_books
5.	tome	1908_books
6.	storybook	1887_books
7.	pop-up_book	1882_books
8.	bestiary	1872_books
9.	catechism	Books_by_Kierkegaard
10.	trade_book	Travel_books

Table 4: Results for the query <$x subclassOf book>

Clearly, the top 9 results are a direct result of personalization. Results at rank 3 through 8 are the years in which Nietzsche's works were published. The result at rank 10 was a "random" result – that is, there were no other results which could be preferred since all other results had the same score.

Rank	NAGA	Personalization
1.	type artist	created "The Potato Easters"
2.	type person	created Sunflowers
3.	type Dutch painter	created "The Starry Night"
4.	isCalled *	created Irises
5.	is Called Vinsent van Qoq	created "Portrait of Dr. Gachet"
6.	isCalled *	type artist
7.	isCalled *	type person
8.	isCalled *	type Dutch painter
9.	isCalled *	isCalled *
10.	isCalled *	is Called Vinsent van Qoq

Table 5: Results for the query "Vincent_van_Gogh $r $x"

Next, we discuss an example pointing out the merits of fact score propagation. Suppose a user showed interest in the works of Picasso (profile 10), and is now searching facts about Vincent van Gogh (<Vincent_van_Gogh $R $x>). Without personalization we find answers such as Vincent van Gogh is an artist on top followed by the synonyms under which he was known in various languages. In contrast, the personalized results present a number of his paintings on top (see Table 5 for the result rankings). This is due to the interest propagation from <Picasso CREATED Chicago_Picasso> to, e.g., <Vincent_van_Gogh CREATED Sunflowers> via the relation CREATED.

In summary, we have shown in this section that not only is personalization highly essential (comparing the ranked result list of NAGA with the personalized list), but also that our techniques are the right way to proceed. We tested our techniques with both simple as well as complex queries. We have reported on a subset of the most interesting queries here.

6. RELATED WORK

Our present work on personalizing the search for knowledge finds related work on personalization in various fields:

in Web search, XML search, as well as in databases. However, to the best of our knowledge we are the first to endeavor to personalize search in a knowledge base. This bears commonalities with personalization in databases as both lack elaborate textual information. At the same time, the knowledge base we consider is schema-free as opposed to databases and XML where a successful personalization system always needs to be adapted to the schema at hand.

In [8], Koutrika et al. present a personalization framework for database users that relies on user profiles which comprise user preferences with respect to atomic query elements such as individual join conditions or selections. At query-time, the preferences most relevant to the query and user are selected from the profile, and integrated to form a new modified database query. This query-rewriting process is inherently hard as, e.g., conflicting constraints need to be handled. Koutrika et al. provide means for handling syntactical conflicts, however, semantical conflicts are schema-dependent and thus not generally addressable.

In [2], the XML personalization system, PIMENTO is presented. In their approach, a user profile is a set of rules of the form (*condition, action, conclusion*). The condition and conclusion parts are XQuery full text, and the action can be *add*, *remove* or *replace*. Whenever a query matches a rule condition, it is re-written accordingly. However, the generation of the rules in the user profile requires the user's active participation. Pan describes an interesting approach in [10] where he proposes query expansion for XML search based on ontological similarities. After initial feedback from the user a query specific ontology is constructed from parts of the global ontology and the query itself, and then used for expansion. In [11, 12] Schenkel and Theobald present a way for structural query expansion based on relevance feedback for XML. [5] is an approach along similar lines based on pseudo feedback and only a portion of the document. XML structure is also considered by Wang et al. in [16]. They propose an extension of Preference XPath for the purposes of personalized MPEG-7 digital libraries.

Jiang et al. [6] introduce a personalization approach for semantic search. Their methodology is similar to ours in that they employ a domain ontology, and reason on user interest in concepts and relations. They consider long-term interest scores based on previous accesses, and perform spreading activation on the concepts present in the initial query result set. By combining scores from the spreading activation with the long-term interest weights, results "close" to the result set that have been accessed before will be ranked highest. By performing spreading activation on the result set instead of the user profile, this approach, however, does not allow to infer interest in concepts related to the ones accessed by the user. Similarly, Sieg et al. [13] present an approach to personalized search that involves building models of user context as ontological profiles by assigning implicitly derived interest scores to existing concepts in a domain ontology. Still both approaches do not tackle the problem of personalization the search on a knowledge graph but consider personalizing document search.

Another differing, still related area is ontological user profiling in recommender systems. For example, Middleton et al. [9] present two experimental systems, Quickstep and Foxtrott, for recommending academic research papers.

7. CONCLUSIONS AND FUTURE WORK

In this paper we presented a personalization approach tailored to the specifics of searching a knowledge base. We developed techniques to infer user interests to create a user profile and then showed how user interest could be incorporated in our model for personalization. Our preliminary experiments examined both entity and fact score propagation and showed promising results.

In ongoing experiments, we are studying how mixing user interest with the original NAGA score affects results. We are also studying how the user-specific background model affects results.

8. ADDITIONAL AUTHORS

9. REFERENCES

[1] S. Amer-Yahia, I. Fundulaki, P. Jain, and L. Lakshmanan. Personalizing xml text search in piment. In *Proc. of VLDB*, 2005.

[2] S. Amer-Yahia, I. Fundulaki, and L. Lakshmanan. Personalizing xml search in pimento. In *Proc. of ICDE*, 2007.

[3] M. Cafarella, C. Re, D. Suciu, and O. Etzioni. Structured querying of web text data: A technical challenge. In *Proc. of CIDR*, 2007.

[4] D. Hiemstra. A probabilistic justification for using tf.idf term weighting in information retrieval. *Inf. Proc. and Management*, 3(2), 2006.

[5] W. Hsu, M. L. Lee, and X. Wu. Path-augmented keyword search for xml documents. In *ICTAI*, 2004.

[6] X. Jiang and A.-H. Tan. Learning and inferencing in user ontology for personalized semantic web services. In *Proc. of WWW*, 2006.

[7] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. Naga: Searching and ranking knowledge. In *Proc. of ICDE*, 2008.

[8] G. Koutrika and Y. Ioannidis. Personalization of queries in database systems. In *Proc. of ICDE*, 2004.

[9] S. E. Middleton, N. R. Shadbolt, and D. C. D. Roure. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, 22(1):54–88, 2004.

[10] H. Pan. Relevance feedback in xml retrieval. In *EDBT Workshops*, 2004.

[11] R. Schenkel and M. Theobald. Feedback-driven structural query expansion for ranked retrieval of xml data. In *Proc. of EDBT*, 2006.

[12] R. Schenkel and M. Theobald. Structural feedback for keyword-based xml retrieval. In *Proc. of ECIR*, 2006.

[13] A. Sieg, B. Mobasher, and R. Burke. Web search personalization with ontological user profiles. In *Proc. of CIKM*, 2007.

[14] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *Proc. of WWW*, 2007.

[15] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proc. of SIGIR*, 2005.

[16] Q. Wang, W.-T. Balke, W. Kiessling, and A. Huhn. P-news: Deeply personalized news dissemination for mpeg-7 based digital libraries. In *Proc. of ECDL*, 2004.

[17] C. Zhai and J. Lafferty. A risk minimization framework for information retrieval. *Inf. Proc. and Management*, 42(1), 2006.

Preferential Publish/Subscribe

Marina Drosou
Dept. of Computer Science
University of Ioannina, Greece
mdrosou@cs.uoi.gr

Evaggelia Pitoura
Dept. of Computer Science
University of Ioannina, Greece
pitoura@cs.uoi.gr

Kostas Stefanidis
Dept. of Computer Science
University of Ioannina, Greece
kstef@cs.uoi.gr

ABSTRACT

In publish/subscribe systems, subscribers express their interests in specific events and get notified about all published events that match their interests. As the amount of information generated increases rapidly, to control the amount of data delivered to users, we propose enhancing publish/subscribe systems with a ranking mechanism, so that only the top-ranked matching events are delivered. Ranking is based on letting users express their preferences on events by ordering the associated subscriptions. To avoid the blocking of new notifications by top-ranked old ones, we associate with each notification an expiration time. We have fully implemented our approach in SIENA, a popular publish/subscribe middleware system.

1. INTRODUCTION

The publish/subscribe paradigm provides loosely coupled interaction among a large number of users of a large-scale distributed system. Users can express their interest in an event via a *subscription* and inject this subscription into the system. The system will then *notify* them whenever some other user generates (or *publishes*) an event that *matches* a previously made subscription. Users that generate such events are called *publishers*, while users that consume the published events are called *subscribers*. All published events that are relevant to at least one of a specific user's subscription will eventually be delivered to this user.

Typically, in publish/subscribe systems, all subscriptions are considered equally important. However, due to the abundance of information, users may receive overwhelming amounts of event notifications. In such cases, users would prefer to receive only a fraction of this information, namely the most interesting to them. For example, assume a user, say John, who is generally interested in drama movies. Specifically, John is more interested in drama movies directed by Tim Burton than drama movies directed by Steven Spielberg. Ideally, John would like to receive notifications about Steven Spielberg drama movies only in case there are no, or not enough, notifications about Tim Burton drama movies.

In this paper, we advocate using some form of ranking among subscriptions, so that users can express the fact that some events are *more important* to them than others. To rank subscriptions, we use preferences. A variety of preference models have been proposed, most of which follow either a quantitative or a qualitative approach. In the *quantitative approach* (e.g. [5, 13]), users employ scoring functions that associate a numeric score with specific data to indicate their interest in it. In the *qualitative approach* (e.g. [8, 12, 11]), preferences between two data items are specified directly, typically using binary preference relations. To express priorities among subscriptions, we first introduce *preferential subscriptions*, that is, subscriptions enhanced with interest scores following the quantitative preference paradigm. Based on the subscription scores, we propagate to users only the notifications that are the most interesting to them. We extend this idea to encompass qualitative preferences.

Based on preferential subscriptions, we introduce a top-k variation of the publish/subscribe paradigm in which users receive only the k most interesting events as opposed to all events matching their subscriptions. Since the delivery of notifications is continuous, we introduce a timing dimension to the top-k problem, since without some notion of freshness, receiving top-ranked notifications would block for ever the delivery of any new, less interesting notifications. To this end, we associate an expiration time with each notification, so that, notifications for old events will eventually die away and let new ones be delivered to the users.

To locate the subscriptions that match a specific event notification efficiently, we adopt a graph-based representation of subscriptions, called *preferential subscription graph*. Subscriptions correspond to nodes in the graph and edges point from more general to more specific subscriptions.

Our prototype implementation, PrefSIENA [3], extends SIENA [6], a popular publish/subscribe middleware system, by including preferential subscriptions and top-k notification delivery. We report some preliminary experimental results that compare the number of notifications delivered by PrefSIENA with respect to the corresponding number in the case of the original SIENA system.

The rest of this paper is structured as follows. In Section 2, we introduce preferential subscriptions, that is, subscriptions augmented with interest scores. We also propose time-valid notifications by associating expiration times with notifications. In Section 3, we focus on how to compute the top-k notifications based on preferential subscriptions and time-valid notifications, while in Section 4, we extend pref-

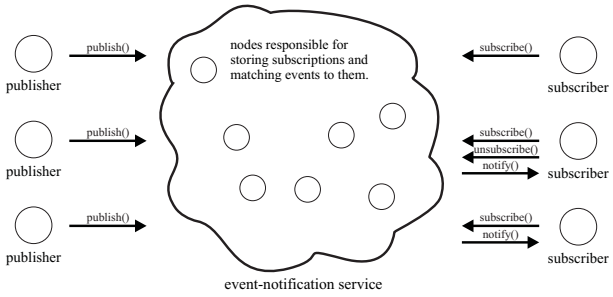

Figure 1: Basic publish/subscribe system.

erential subscriptions to encompass qualitative preferences. In Section 5, we present our evaluation results. Section 6 describes related work and finally, Section 7 concludes the paper with a summary of our contributions.

2. PREFERENTIAL MODEL

In this section, we first describe a typical form of notifications and subscriptions used in publish/subscribe systems. Then, we introduce an extended version of subscriptions that include the notion of preferences. Also, we extend the definition of notifications to include the notion of time validity.

2.1 Publish/Subscribe Preliminaries

A publish/subscribe system is an event-notification service designed to be used over large-scale networks, such as the Internet. Generators of events, called *publishers*, can publish event notifications to the service and consumers of such events, called *subscribers*, can subscribe to the service to receive a portion of the published notifications. Publishers can publish notifications at any time. The notifications will be delivered to all interested subscribers at some point in the future.

Architecture: In general, a publish/subscribe system [9] consists of three parts: (i) the publishers that provide events to the system, (ii) the subscribers that consume these events and (iii) an event-notification service that stores the various subscriptions, matches the incoming event notifications against them and delivers the notifications to the appropriate subscribers. As shown in Figure 1, the event-notification service provides a number of primitive operations to the users. The **publish**() operation is called by a publisher whenever it wishes to generate a new event. The **subscribe**() operation is called by a subscriber whenever it wishes to express a new interest. An **unsubscribe**() operation is usually also provided to cancel previous subscriptions. The event-notification service can use the **notify**() operation whenever it wants to deliver a notification to a subscriber. An event-notification service can be implemented using a centralized or a distributed architecture, that is, we may have one or a set of servers responsible for the process of matching notifications to subscriptions.

Notifications: We use a generic way to form notifications, similar to the one used in [6, 10]. In particular, notifications are sets of typed attributes. Each notification consists of an arbitrary number of attributes and each attribute has a type, a name and a value. Attribute types belong to a predefined set of primitive types, such as "integer" or "string". Attribute names are character strings that take values according to their type. An example notification about a movie is shown in Figure 2.

string title	=	LOTR: The Return of the King
string director	=	P. Jackson
time release_date	=	1 Dec 2003
string genre	=	fantasy
integer oscars	=	11

Figure 2: Notification example.

string director	=	P. Jackson
time release_date	>	1 Jan 2003

Figure 3: Subscription example.

DEFINITION 1 (NOTIFICATION). *A notification n is a set of typed attributes $\{a_1, \ldots, a_p\}$, where each a_i, $1 \leq i \leq p$, is of the form ($a_i.type$ $a_i.name = a_i.value$).*

Subscriptions: Subscriptions are used to specify the kind of notifications users are interested in. Each subscription consists of a set of constraints on the values of specific attributes. Each attribute constraint has a type, a name, a binary operator and a value. Types, names and values have the same form as in notifications. Binary operators may include common operators such as $=, \neq, <, >, \leq, \geq$, $substring$, $prefix$ and $suffix$. An example subscription is depicted in Figure 3.

DEFINITION 2 (SUBSCRIPTION). *A subscription s is a set of attribute constraints $\{b_1, \ldots, b_q\}$, where each b_i, $1 \leq i \leq q$, is of the form ($b_i.type$ $b_i.name$ θ_{b_i} $b_i.value$), $\theta_{b_i} \in \{=, <, >, \leq, \geq, \neq, substring, prefix, suffix\}$.*

Matching notifications to subscriptions: Intuitively, we can say that a notification n *matches* a subscription s, or alternatively a subscription s *covers* a notification n, if and only if every attribute constraint of s is satisfied by some attribute of n. Formally:

DEFINITION 3 (COVER RELATION). *Given a notification n of the form $\{a_1, \ldots, a_p\}$ and a subscription s of the form $\{b_1, \ldots, b_q\}$, s covers n if and only if $\forall\ b_i \in s$, $\exists\ a_j \in n$ such that $b_i.type = a_j.type$, $b_i.name = a_j.name$ and it holds $((a_j.value)\ \theta_{b_i}\ (b_i.value))$, $1 \leq i \leq p$, $1 \leq j \leq q$.*

A notification n is delivered to a user if and only if the user has submitted at least one subscription s, such that s covers n. For example, the subscription of Figure 3 covers the notification of Figure 2 and therefore, this notification will be delivered to all users who have submitted this subscription.

2.2 Preferential Subscriptions

In this paper, we extent the publish/subscribe paradigm to incorporate ranking capabilities. Assuming that each user has defined a set of preferences, then this user should receive a newly published notification, if and only if, the notification describes an event that is more preferable to the user than any previously received event. To express preferences along with subscriptions, we can follow either the quantitative or the qualitative approach. For simplicity reasons, we first consider the quantitative preference model, while in Section 4, we apply qualitative preferences.

A *preferential subscription* is a subscription enhanced with a numeric score. The higher the score, the more interested

string director	=	P. Jackson	0.7
date release_date	>	1 Jan 2003	

Figure 4: Preferential subscription example.

the user is in notifications covered by this specific subscription. These scores can have any real value. We assume here that they take values within the range $[0, 1]$. An example of a preferential subscription is shown in Figure 4.

DEFINITION 4 (PREFERENTIAL SUBSCRIPTION). *A preferential subscription ps_i^X, submitted by user X, is of the form $ps_i^X = (s_i, score_i^X)$, where s_i is a subscription and $score_i^X$ is a real number within the range $[0, 1]$.*

Assuming that a user X defines a set of preferential subscriptions P^X, we use the user's preferential subscriptions to rank the published notifications and deliver to the user only the top-k notifications, i.e. the k highest ranked ones (where k is a user-defined parameter). We define the score of a notification to be the largest among the scores of the subscriptions that cover it:

DEFINITION 5 (NOTIFICATION SCORE). *Assume a notification n, a user X and the set P^X of the user's preferential subscriptions. Assume further the set $P_n^X = \{(s_1, score_1^X), \ldots, (s_m, score_m^X)\}$, $P_n^X \subseteq P^X$, for which s_i covers n, $1 \leq i \leq m$. The notification score of n for X is equal to $sc(n, X) = max\ \{score_1^X, \ldots, score_m^X\}$.*

A newly published notification n is delivered to a user X, if and only if, it is covered by some subscription s previously issued by X and X has not already received k notifications more preferable to n. A notification n_1 is more preferable for user X to a notification n_2, if and only if, it has a higher notification score for X than n_2.

In general, we assume that scores are indicators of positive interest, thus, we use the maximum value of the corresponding subscriptions. One could argue for other ways of aggregating the scores, for instance using the mean, minimum or a weighted sum. Yet, an alternative way would be to use only the scores of a subset of P_n^X, namely the *most specific* subscriptions. A similar notion for preferences is introduced in [15]. For example, assume the notification of Figure 2 and the preferential subscriptions $(\{genre = fantasy\}, 0.7)$ and $(\{genre = fantasy, director = P.\ Jackson\}, 0.6)$ (for ease of presentation we omit the type of each attribute). The second subscription is more specific than the first one, in the sense that in the second subscription the user poses an additional, more specific requirement to movies than in the first one, and so, the score of the first one should be ignored. Formally, a subscription $s \in P_n^X$ is a most specific one if no other subscription in P_n^X covers it (see Definition 7).

We leave as future work a user study to evaluate the appropriateness of the different methods for assigning scores. In general, all such methods may increase the complexity of the process of matching notifications to subscriptions, since in traditional publish/subscribe, for matching to be completed successfully, it suffices to find just one subscription that covers the notification, whereas for computing the notification score, we may need to locate all covering subscriptions.

2.3 Time-Valid Notifications

In a publish/subscribe system, where new event notifications are constantly produced, the following problem may arise: it is possible for very old but highly preferable notifications to prevent newer notifications from reaching the user. Specifically, after receiving k very highly preferable notifications, the user will receive no new ones unless they are ranked higher, something that may not be desirable.

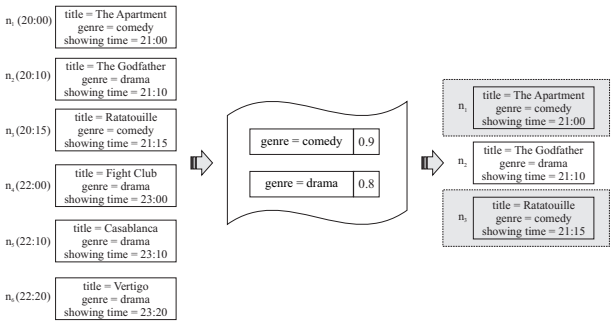

Figure 5: Top-2 notifications for a single user at 22:30 (no expiration time used).

For instance, consider the example of Figure 5. For simplicity, we assume a single user, say John, who has defined the following preferential subscriptions for movies: John has assigned score 0.9 to comedies and score 0.8 to dramas. Assume that a movie theater generates the notifications $n_1, n_2, \ldots, n_6$ of Figure 5 in that order and that John is interested in the top-2 results. n_1 will be delivered to John, since it is the first notification that is covered by his subscriptions. n_2 will also be delivered, since it is the second best result seen up to this moment. n_3 is equally preferred to n_1 and will therefore also be delivered to John (replacing n_2 in the current top-2 results). Since n_4, n_5 and n_6 are less preferable to the current top-2 results, none of them will be delivered to John. Assuming that notifications are published one hour prior to the showing time, if John checks his top-2 results at 22:30 he will only find movies that he can no longer watch (the top-2 results at 22:30 are marked with gray color), even though other interesting movies that start at 23:00 have been published.

To overcome this problem, we need to define the subset of notifications over which the top-k notifications for each user will be located. One solution would be to split time in periods of duration T and at each time instance, deliver a notification to the user, if the user has not already received, during the current period, k notifications with higher scores. However, since top-k computation starts anew in the beginning of each period, the rank of events received by the user may end up being rather arbitrary. For example, high-ranked notifications appearing in periods with many other high-ranked ones may not be delivered to the user, whereas low-ranked publications appearing in periods with a small number of high-ranked ones may be delivered.

A more general approach is to associate each published notification n with an expiration time $n.exp$. The notification is considered *valid* only while $n.exp$ has not expired. The top-k results for each user are defined over the subset of valid notifications. This way, older notifications which have expired do not prevent valid ones from reaching the user even if they are more preferable than those. Considering the previous example, assume that each notification expires at the showing time of the corresponding movie (see Figure 6). n_1, n_2 and n_3 will be delivered to the user as

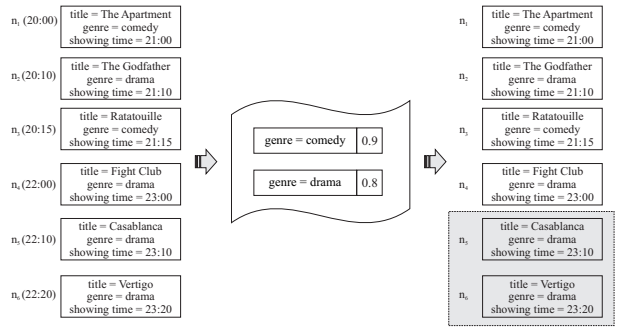

Figure 6: Top-2 notifications for a single user at 22:30 (with expiration time used).

before. By the time n_4 is published (22:00), n_1, n_2 and n_3 have expired and therefore, n_4 will also be delivered to the user. n_5 and n_6 will be delivered as well, since they are equally preferred as the notifications in the top-2 at the time of their publication. Notice that the periodic approach is a special case of the expiration-time one. By setting the expiration time of each notification equal to the ending time of the current period, we achieve the same result as with the periodic approach.

Based on the assumption that published notifications are valid only for a specific time period, next we define which ones belong to the top-k notifications for a user.

DEFINITION 6 (TOP-k NOTIFICATIONS). *Assume a user X and P^X the set of X's preferential subscriptions. A notification n published at time t belongs to the top-k notifications of X, if and only if, n is covered by at least one subscription s appearing in a preferential subscription $ps \in P^X$ and X has not already received k notifications $n_1, \ldots, n_k$ with $n_i.exp > t$ and $sc(i, X) > sc(n, X)$, $1 \leq i \leq k$.*

An alternative way to set the expiration time for a notification would be to let the user define a refresh time along with each subscription. This can be also expressed through defining appropriate values of the expiration time for notifications as follows. Assuming that a notification n is covered by a user subscription s associated with a refresh time r, then the expiration time of n could be set to $t + r$, where t is the time that n is sent to the user. Note that in this approach, a specific notification does not have a single expiration time but instead, it is associated with a different one for each user.

3. RANKING IN PUBLISH/SUBSCRIBE

In this section, we introduce a *preferential subscription graph* for organizing our preferential subscriptions. We also present an algorithm for computing the top-k results and discuss the server topology.

3.1 Preferential Subscription Graph

To reduce the complexity of the matching process between notifications and subscriptions, it is useful to organize the subscriptions using a graph. We use preferential subscriptions to construct a directed acyclic graph, called *preferential subscription graph*, or PSG. To form such graphs, we use the cover relation between subscriptions defined as follows.

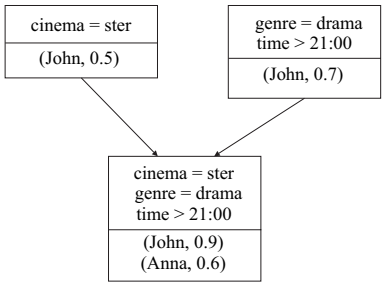

Figure 7: Preferential subscription graph example.

DEFINITION 7 (COVER BETWEEN SUBSCRIPTIONS). *Given two subscriptions s_i and s_j, s_i covers s_j, if and only if, for each notification n such that s_j covers n, it holds that s_i covers n.*

For example, the subscription $\{genre = fantasy, director = P.\ Jackson\}$ covers the subscription $\{genre = fantasy\}$.

In a preferential subscription graph, nodes correspond to subscriptions and edges to cover relations between subscriptions. Assume the set P of all preferential subscriptions, i.e. the preferential subscriptions defined by all users. For each subscription $s_i \in S_P$, where S_P is the set of all subscriptions in P, we maintain a set of pairs, called *score set*, of the form $(j, score_i^j)$, where j is a user and $score_i^j$ is the numeric score that j has assigned to s_i. A subscription s_i is associated with the pair $(j, score_i^j)$, if and only if, a preferential subscription $ps_i^j = (s_i, score_i^j)$ exists in P. Next, we define formally the score set of a subscription.

DEFINITION 8 (SCORE SET). *Assume a set of users U, a set of preferential subscriptions P, and S_P the set of all subscriptions in P. For each $s_i \in S_P$, the score set is the set $W_i = \{(j, score_i^j) \mid (s_i, score_i^j) \in P\}$.*

Having defined the score set of a specific subscription, we now define the preferential subscription graph.

DEFINITION 9. (PREFERENTIAL SUBSCRIPTION GRAPH). *Let P be a set of preferential subscriptions and S_P the set of all subscriptions in P. A Preferential Subscription Graph $PSG_P(V_P, E_P)$ is a directed acyclic graph, where for each different $s_i \in S_P$, there exists a node v_i, $v_i \in V_P$, of the form (s_i, W_i), where W_i is the score set of s_i . Given two nodes v_i, v_j, there exists an edge from v_i to v_j, $(v_i, v_j) \in E_P$, if and only if, s_i covers s_j and there is no node v'_j such that s_i covers s'_j and s'_j covers s_j.*

For example, assume two users, John and Anna, who express the following preferential subscriptions: John gives to subscription $s_1 = \{cinema = ster, genre = drama, time > 21{:}00\}$ score 0.9, to $s_2 = \{genre = drama, time > 21{:}00\}$ score 0.7 and to $s_3 = \{cinema = ster\}$ score 0.5. Similarly, Anna assigns to s_1 score 0.6. For the above preferential subscriptions, the graph of Figure 7 is constructed.

The preferential subscription graph resembles the *filters poset* data structure proposed in [6]. Whereas the filters poset represents a partially ordered set of subscriptions, the preferential subscription graph is based on subscriptions enhanced with interest scores.

3.2 Forwarding Notifications

To show how the top-k results for each user are computed, we first assume a single server maintaining a preferential

subscription graph PSG. In the next section, we generalize our approach for more servers. This single server acts as an access point for all subscribers and publishers. Although publish/subscribe systems are typically stateless, in the sense that they do not maintain any information about previous notifications, here, we need to maintain some information about previously sent top-ranked notifications. The server maintains a list of k elements for each of the subscribers (users) that are connected to it. These lists contain elements of the form $(score, expiration)$ where $score$ is a numeric value and $expiration$ is a time field. The $score$ part of such a pair represents the score of a notification that has already been delivered to the corresponding user and expires at time $expiration$. Only the scores corresponding to the top-k most preferable valid notifications that have been already sent to the users appear in these lists.

All lists are initially empty. Whenever the server receives a notification n, it walks through its PSG to find all subscriptions that cover n. For each subscriber j associated with at least one of these subscriptions, a score $sc(n, j)$ is computed: assuming that m subscriptions $s_1, s_2, \ldots, s_m$ submitted by j cover n, then $sc(n, j) = max\ \{score_1^j, score_2^j, \ldots, score_m^j\}$. After that, the corresponding list, denoted $list^j$, is checked and all elements which have expired are removed. If $list^j$ contains less than k elements, n is forwarded to j and the pair $(sc(n, j), n.exp)$ is added to the list, where $n.exp$ is the expiration time of n. Otherwise, n is forwarded to j only if $sc(n, j)$ is greater or equal to the score of the element with the minimum score in the list. In this case, this element is replaced by $(sc(n, j), n.exp)$. Note that, a more recent notification equally important to an older one is forwarded to the user to favor fresh data over equally-ranked old ones. The process described above is summarized in the *Forward Notification Algorithm* shown in Algorithm 1.

Algorithm 1 Forward Notification Algorithm

Input: A notification n and a preferential subscription graph PSG.
Output: The set of subscribers $ResSet$ n will be forwarded to.

```
1: Begin
2: ResSet = ∅;
3: tmpW = ∅;          /* temporary score set */
4: for all nodes v_i in PSG do
5:    if s_i covers n then
6:       tmpW = tmpW ∪ W_i;
7:    end if
8: end for
9: for all subscribers j that appear in tmpW do
10:   sc(n, j) = max{score_1^j, ..., score_{m^j}^j}, where (j, score_i^j)
      ∈ tmpW, 1 ≤ i ≤ m^j;
11:   for all elements i in list^j do
12:      if i has expired then
13:         remove i from list^j;
14:      end if
15:   end for
16:   if list^j contains less than k elements then
17:      add (sc(n, j), n.exp) to list^j;
18:      ResSet = ResSet ∪ j;
19:   else
20:      find the element i of list^j with the minimum score;
21:      if sc(n, j) > i.score then
22:         remove i from list^j;
23:         add (sc(n, j), n.exp) to list^j;
24:         ResSet = ResSet ∪ j;
25:      end if
26:   end if
27: end for
28: return ResSet;
29: End
```

Next, we prove the completeness and correctness of Algorithm 1. First, we will show that if a notification n belongs to the top-k results of user j, then it will be forwarded to j. Assume for the purpose of contradiction, that such a notification is not forwarded to j. Let $sc(n, j)$ be the score of n for j. Since n is not forwarded to j, there exist k valid notifications $n_1, \ldots, n_k$ with scores $sc(n_1, j), \ldots, sc(n_k, j)$ such that $sc(n_i, j) > sc(n, j)$, $1 \leq i \leq k$. This means that n does not belong to the top-k results of user j, which violates our assumption. Next, we proceed with showing that if a notification n is forwarded to j, then it belongs to the user's top-k results. For the purpose of contradiction, assume that n does not belong to the user's top-k results. This means that there exist k valid notifications $n_1, \ldots, n_k$ with scores $sc(n_1, j), \ldots, sc(n_k, j)$ such that $sc(n_i, j) > sc(n, j)$, $1 \leq i \leq k$. Therefore, according to Algorithm 1 (line 21), n will not be forwarded to j, which is a contradiction.

Note that it is not necessary to walk through all nodes of the preferential subscription graph to locate the subscriptions that cover a specific notification. We may safely ignore a node v with subscription s for which there is no other node v' with subscription s', such that s' covers s and at the same time s' covers n. This way, entire paths of the graph can be pruned and not used in the matching process.

3.3 Hierarchical Topology of Servers

An event-notification service can be implemented over various architectures. At one extreme, a centralized approach can be followed, e.g. [10]. In this case, a single server gathers all subscriptions and notifications and carries out the matching process. However, due to the nature of such systems, where participants are physically distributed across the globe, a distributed architecture is more scalable. When more than one server exists in the network, each server runs Algorithm 1 for its own preferential subscription graph. Notifications are propagated among servers based on the server topology. The servers of the system are responsible for collecting all the published notifications and carrying out the selection process, i.e. delivering each notification only to the subscribers that have declared their interest to it.

We consider a hierarchical topology, where the servers that implement the event-notification service are connected to each other to form a hierarchy. Each publisher and subscriber is connected to one of the servers in the hierarchy.

Furthermore, we wish to organize the participants of the network in an efficient way, i.e. in a way that will reduce the number of messages exchanged between the servers and the complexity of the maintained data structures. One way to achieve this is by placing subscribers with similar subscriptions nearby in the hierarchy, so that the notifications covered by those subscriptions need to be propagated only toward this part of the hierarchy.

While in most publish/subscribe systems, new subscribers randomly select a server to connect to, in our approach, when a new subscriber enters the network it probes a number of servers and chooses one of them according to a number of criteria:

- (*Criterion 1*) The number of new nodes added to the highest level of the server's preferential subscription graph. The smaller the number of such nodes, the

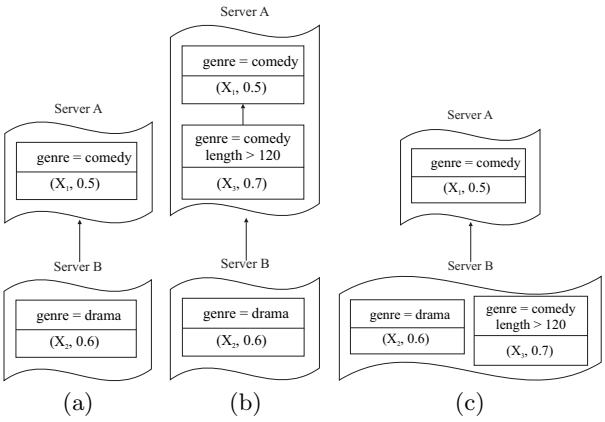

Figure 8: Clustering.

fewer the additional notifications that should be propagated to the server in the future.

- (*Criterion 2*) The number of nodes in the server's preferential subscription graph. The fewer the nodes in the graph, the lower the complexity of searching it.

A new subscriber X chooses a server to subscribe according to the above criteria. For instance, X may first use *Criterion 1*, and in case of a tie, *Criterion 2*. For example, consider the case of Figure 8a where there are two servers, Server A and Server B, both already storing some user subscriptions from subscribers X_1 and X_2. Assume that a new subscriber X_3 wishes to insert a new preferential subscription ($\{genre = comedy, length > 120\}$, 0.7) to the system. If X_3 chooses Server A to subscribe, the result will be the one shown in Figure 8b. If X_3 chooses Server B, the result will be the one shown in Figure 8c. Using the first criterion, X_3 will choose to join Server A because in this case no new nodes will be added to the highest level of the PSG of Server A and thus, no new message traffic will be generated (except from the messages sent from Server A to X_3).

4. SUBSCRIPTIONS USING A QUALITATIVE MODEL

Preferential subscriptions as defined in Section 2.2 exploit the notion of quantitative preferences. Here, we discuss how to express preferential subscriptions using a qualitative preference model.

Assume that a user X provides a set of subscriptions S_P^X. To define choices between subscriptions, X expresses priority conditions of the form $s_i \succ s_j$, s_i, $s_j \in S_P^X$, to denote that s_i is preferred to s_j for X. Let C^X be the set of priority conditions expressed by user X, i.e. $C^X = \{(s_i \succ s_j) \mid s_i, s_j \in S_P^X\}$. To extract the most preferable subscriptions C^X, we use the winnow operator [8]. In particular, the first application of the winnow operator returns the set $win_X(1)$ of subscriptions $s_i \in S_P^X$ such that $\forall s_i \in win_X(1)$ there is no $s_j \in S_P^X$ with $s_j \succ s_i$. If we would like to retrieve further the most preferable subscriptions after the ones included in $win_X(1)$, we apply the winnow a second time. $win_X(2)$ consists of the subscriptions $s_i \in (S_P^X - win_X(1))$ such that $\forall s_i \in win_X(2)$ there is no subscription $s_j \in (S_P^X - win_X(1))$ with $s_j \succ s_i$. The winnow operator may be applied until all subscriptions are returned. To locate the subscriptions that belong to a specific winnow result set, we define the *multiple level winnow operator*.

Definition 10. (Multiple Level Winnow Operator). *Assume a user X and let S_P^X be the set of X's subscriptions. Let C^X be the set of priority conditions of X, the multiple level winnow operator at level l, $l > 1$, returns a set of subscriptions, $win_X(l)$, consisting of the subscriptions $s_i \in S_P^X - \cup_{q=1}^{l-1} win_X(q)$ such that $\forall\ s_i \in win_X(l)\ \nexists s_j \in S_P^X - \cup_{q=1}^{l-1} win_X(q)$ with $(s_j \succ s_i) \in C^X$.*

To compute the top-k results for each user when priority conditions between subscriptions are specified, we modify the process described in Algorithm 1 as follows. Again, each server maintains a list of k elements for each user that is connected to it. These lists now contain elements of the form (pos, $expiration$) where pos represents a position value and $expiration$ is a time field. The pos value denotes the winnow level that a subscription that covers a notification which has already been delivered to the user belongs to, and $expiration$ the time instance the notification expires. Again, only the elements corresponding to the top-k most preferable valid notifications that have been already sent to the users appear in these lists. In this work, we assume that there are no conflicting priority conditions.

Whenever the server receives a notification n, it walks through its PSG to find all subscriptions that cover n. For each subscriber j associated with at least one of these subscriptions, a value $rank(n, j)$ is calculated: assuming that m subscriptions $s_1, s_2, \ldots, s_m$ submitted by j cover n, then $rank(n, j) = min\ \{level_1^j, level_2^j, \ldots, level_m^j\}$, where $level_i^j$ denotes the winnow level that the subscription s_i belongs to. In the following, the corresponding list, $list^j$, is checked and all expired elements are removed. If $list^j$ contains less than k elements, n is forwarded to j and the pair ($rank(n, j)$, $n.exp$) is added to the list, where $n.exp$ is the expiration time of n. Otherwise, n is forwarded to j only if $rank(n, j)$ is less or equal to the rank value of the element with the maximum rank in the list. In this case, this element is replaced by $(rank(n, j), n.exp)$.

5. EVALUATION

To evaluate our approach, we have extended the SIENA event notification service [4], a multi-threaded publish/subscribe implementation, to include preferential subscriptions. We refer to our implementation as PrefSIENA. Our source code is available for download at [3].

System Description: To evaluate the performance of our model, we use a real movie-dataset [2], which consists of data derived from the Internet Movie Database (IMDB) [1]. The dataset contains information about 58788 movies. For each movie the following information is available: title, year, budget, length, rating, MPAA and genre.

string title	=	LOTR: The Return of the King
integer year	=	2003
integer length	=	251
integer rating	=	9
string mpaa	=	PG-13
string genre	=	Action

Figure 9: Generated notification.

Each publisher randomly selects m_P numbers from 1 to 58788. For each of the corresponding m_P movies, the pub-

lisher creates a new notification consisting of the title, year, length, rating, MPAA and genre of the movie. An example of such a notification can be seen in Figure 9. Each subscriber generates m_S subscriptions and each subscription is generated independently from the others. We randomly select a number of the available attributes to appear in a subscription. The value of each attribute can be generated using either a uniform (i.e. all values are equally preferable) or a zipf distribution (i.e. some values are more popular) according to the values appearing in the dataset. In both cases, a subscription is associated with a numeric score uniformly distributed in $[0, 1]$. Subscription examples can be seen in Figure 10.

string genre	=	Romance	0.3
integer length	>	120	
string mpaa	=	PG-13	

string genre	=	Drama	0.6
integer length	>	100	
integer year	<	1980	
integer rating	>	6	

Figure 10: Generated subscriptions.

Experiments: To run our experiments, we assume a network in which each computer node can act as a publisher, subscriber or server. A combination of these roles is also possible. The servers are organized in a hierarchical topology while clients (i.e. publishers and subscribers) can be connected to any one of the servers. Each involved client executes a series of service requests. More specifically, each publisher generates a number of notifications and injects them into the network. All notifications expire after time τ of their publication. Each subscriber generates a number of subscriptions and chooses a server to connect to and subscribe. After that, each subscriber waits for notifications to arrive.

In general, the number of delivered notifications depends on the covering relations between the various subscriptions and published notifications, the scores associated with these subscriptions and the order in which notifications are generated. The notification receipt rate for each individual user can be fine-tuned by letting the user define appropriate values for refreshing the subscriptions (so that the expiration times of the corresponding notifications are set accordingly) and by selecting k.

First, we measure the number of notifications delivered to a specific subscriber using PrefSIENA as a function of the number k of the top results the subscriber is interested in. We run this experiment with 100 matching events and for expiration time τ equal to $15t$ and $20t$, where $t = 500ms$ is the time length between the generation of two notifications. Note that t refers to real, and not simulated, time. We consider the following two scenarios. In the first one, *scenario 1*, most of the notifications with higher scores for the user are published early, while in the second one, *scenario 2*, notifications with higher scores arrive towards the end. We observe that in the first scenario the user receives fewer notifications than in the second one (Figure 11). This happens because in the first scenario, where notifications with higher scores arrive first, many of the notifications with lower scores cannot enter the top-k results, until some of the first ones expire. In the second scenario, however, the user receives both the

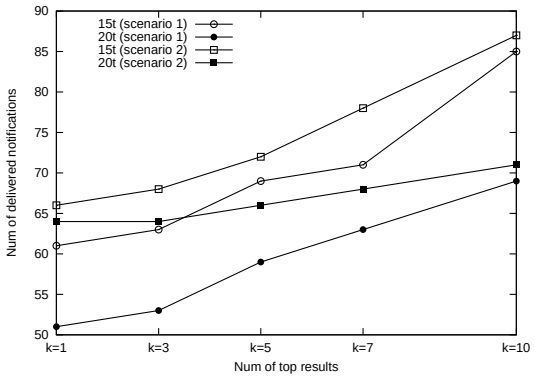

Figure 11: Number of delivered notifications for various values of τ and k.

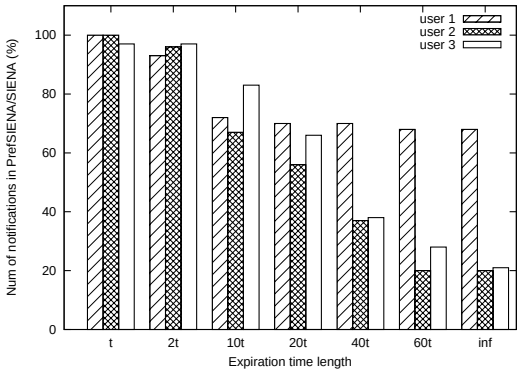

Figure 12: Percentage of delivered notifications for various expiration times (from $t = 500ms$ to infinity)

notifications with the lower scores that arrive first and the notifications with higher scores that arrive later. In both scenarios, the number of delivered notifications decreases with the increase of the expiration time. This happens because notifications with higher scores for the user remain in the top-k results for a longer time period and prevent notifications with lower scores from reaching the user.

Furthermore, we measure the number of notifications delivered to a number of different users in the following cases: (i) using SIENA, (ii) using PrefSIENA with no expiration time for notifications and (iii) using PrefSIENA with a number of different expiration times. The number of published events is 200 and their expiration time τ takes values from t to $60t$, where $t = 500ms$. All subscribers submit 5 different subscriptions and are interested in the top-1 result. We select to show results for three users according to the percentage of notifications that are covered by their subscriptions. The subscriptions of *user 1* cover 51% of the generated notifications. For *user 2* and *user 3*, the percentage values are 27% and 15% respectively. In Figure 12, we depict the results for these three users. We count the percentage of delivered notifications in PrefSIENA over the number of notifications in SIENA. By varying the expiration time, we can achieve different notification receipt rates. This rate depends on the scores of users subscriptions, and also, on the specific time that various notifications in the top-k results expire, as shown by the previous experiment.

We also experimented with using the clustering criteria described in Section 3.3 during bootstrapping. We observed a 27% reduction on average of the total messages exchanged between nodes of the system, even though in this case we have an overhead of extra messages during bootstraping.

6. RELATED WORK

Although there has been a lot of work on developing a variety of publish/subscribe systems, there has been only little work on the integration of ranking issues into publish/subscribe. Recently, [18] considers the case of continuous queries in distributed systems. In this approach, only a subset of publishers provide notifications for a specific query. These publishers are selected according to the similarity of their past publications to the query. Similarity is computed via IR techniques. In [17], user preferences are employed to deliver newly added documents of a digital library to the users. Next, we discuss work related to publish/subscribe systems and preferences.

Publish/Subscribe Systems: The publish/subscribe paradigm can be applied to a number of different architectures. The naive approach is to gather all subscriptions and events to a specific node. This node will be responsible for managing subscriptions, matching the incoming events against them and notifying the appropriate subscribers. This is a centralized approach (e.g. [10]). Often, in publish/subscribe systems the number of participating nodes becomes very large. For scalability reasons, a distributed architecture seems to be more suitable. In this case, the event service is implemented via a network of interconnected servers who act as a middle level for the communication of publishers and subscribers. Various distributed architectures such as hierarchical [6] and DHT-based [7] ones, have been proposed.

There are two widely used methods for users to express their subscriptions: the topic-based method and the content-based one. In the *topic-based* method (such as [14]) there are a number of predefined event topics, usually identified by keywords. Published events are associated with a number of topics. Users can subscribe to a number of individual topics and receive all events associated with at least one of these topics. In the *content-based* method [6, 7], such as the one used in this paper, the classification of the published events is based on their actual content. Users express their subscriptions through constraints which identify valid events. An event matches a subscription, if and only if, it satisfies all of the subscription's constraints. In general, the content-based method offers greater expressiveness to subscribers but is harder to implement.

Preferences: The research literature on preferences is extensive. In general, there are two different approaches for expressing preferences: a quantitative and a qualitative one. In the *quantitative approach* (such as [5, 13, 16]), preferences are expressed indirectly by using scoring functions that associate numeric scores with data items. In the *qualitative approach* (e.g. [8, 12, 11]), preferences between data items are specified directly, typically using binary preference relations.

7. CONCLUSIONS

To control the amount of data delivered to users in publish/subscribe systems, we extend such systems to incorporate ranking capabilities. In particular, in this paper, we address the problem of ranking notifications based on preferential subscriptions, that is, user subscriptions augmented with interest scores. To maintain the freshness of data delivered to users, we associate expiration times with notifications. We organize preferential subscriptions in a graph and utilize it to forward notifications to users. We have fully implemented our approach in SIENA, a popular publish/subscribe middleware system.

There are many directions for future work. One is considering alternative approaches for achieving timeliness, such as computing top-k results over sliding windows of notifications. Among our future plans is also studying a weighting scheme based on both time and relevance for ranking notifications. Finally, the focus of this paper has been on enhancing the expressiveness of publish/subscribe systems. Besides expressiveness, performance is also central in such large-scale dynamic systems. In this respect, we plan to consider additional topologies besides the hierarchical one used in this work.

8. REFERENCES

[1] *The Internet Movie Database*. http://www.imdb.com.
[2] *Movies dataset*. http://had.co.nz/data/movies.
[3] *PrefSIENA*. http://www.cs.uoi.gr/∼mdrosou/PrefSIENA.
[4] *SIENA*. http://serl.cs.colorado.edu/∼serl/dot/siena.html.
[5] R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. *SIGMOD Rec.*, 29(2):297–306, 2000.
[6] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Trans. on Computer Syst.*, 19:332–383, 2001.
[7] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE JSAC*, 20(8):1489–1499, 2002.
[8] J. Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.
[9] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.
[10] F. Fabret, A. H. Jacobsen, F. Llirbat, J. a. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. *SIGMOD Rec.*, 30(2):115–126, 2001.
[11] P. Georgiadis, I. Kapantaidakis, V. Christophides, E. M. Nguer, and N. Spyratos. Efficient rewriting algorithms for preference queries. In *ICDE*, pages 1101–1110, 2008.
[12] W. Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322, 2002.
[13] G. Koutrika and Y. Ioannidis. Personalized queries under a generalized preference model. In *ICDE*, pages 841–852, 2005.
[14] T. Milo, T. Zur, and E. Verbin. Boosting topic-based publish-subscribe systems with dynamic clustering. In *SIGMOD*, pages 749–760, 2007.
[15] K. Stefanidis and E. Pitoura. Fast contextual preference scoring of database tuples. In *EDBT*, pages 344–355, 2008.
[16] K. Stefanidis, E. Pitoura, and P. Vassiliadis. Adding context to preferences. In *ICDE*, pages 846–855, 2007.
[17] Q. Wang, W.-T. Balke, W. Kießling, and A. Huhn. P-news: Deeply personalized news dissemination for mpeg-7 based digital libraries. In *ECDL*, pages 256–268, 2004.
[18] C. Zimmer, C. Tryfonopoulos, K. Berberich, G. Weikum, and M. Koubarakis. Node Behavior Prediction for Large-Scale Approximate Information Filtering. In *LSDS-IR*, 2007.

A Context-Oriented Synchronization Approach

Dirk Draheim
Software Competence Center Hagenberg
Softwarepark 21
4232 Hagenberg, Austria
dirk.draheim@scch.at

Christine Natschläger
Software Competence Center Hagenberg
Softwarepark 21
4232 Hagenberg, Austria
christine.natschlaeger@scch.at

ABSTRACT

Synchronization gained great importance in modern applications and allows mobility in the context of information technology. Users are not limited to one computer any more, but can take their data with them on a laptop. Two common architectures have been developed recently, the *Data-Centric Architecture* as well as the *Service-Oriented Architecture*. This paper compares two existing technologies for the implementation of a mobile client and introduces a new approach, developed based on the requirements of a major insurance company, the *Context-Oriented Architecture*. This approach allows detection and resolution of conflicts within the context in which the objects were changed, while still ensuring data correctness and consistency. Therefore two new synchronization concepts are introduced: the *synchronization of complex objects* and *dialogue-sensitive synchronization*. An application implementing this approach has been realized and successfully deployed.

1. INTRODUCTION

The *context-oriented synchronization approach* introduced in this paper has been developed for the *PreVolution* project, executed by the *Software Competence Center Hagenberg* (SCCH) and the *Institut Fuer Anwendungsorientierte Wissensverarbeitung* (FAW) on behalf of the *Austrian Social Insurance Company for Occupational Risks* (AUVA).

Approximately 3 million employed people and 1.3 million school children and students are by law insured by this company. The AUVA takes care of victims of occupational accidents and diseases, a major goal therefore being the prevention of such accidents and diseases. This is the domain where PreVolution is settled.

The aim of PreVolution is to support consultants visiting the companies for advice and physical examination of their employees. The consultants need to synchronize data available at the AUVA onto their laptops to extend and modify the data at the company site. Since several consultants can visit the same company simultaneously conflicts are possible and should be presented to the consultant for resolution. Additionally, the synchronization process has to perform several tasks like business processes during synchronization, changing data as well as unique-constraint violation recognition. Based on these requirements, using a standardized synchronization concept was not possible and an own synchronization approach had to be implemented.

Therefore, this paper covers the challenges of implementing an own synchronization process and contributes two new synchronization concepts:

- Synchronization of Complex Objects
- Dialogue-Sensitive Synchronization

After an introduction of these concepts in Chap. 2 and a comparison with other research work in Chap. 3, we will discuss a *data-centric* as well as a *service-oriented synchronization architecture* in Chap. 4, followed by the new approach of a *context-oriented architecture*. Chap. 5 will describe the *implementation* of a *context-oriented synchronization* in detail. Chap. 6 will present two existing Microsoft technologies for the implementation of a mobile client. This will be the *Smart Client Offline Application Block* on the one hand and the *Synchronization Services for ADO.NET* on the other. Finally, Chap. 7 will present the most important *lessons learned*.

VLDB '08, August 24-30, 2008, Auckland, New Zealand

2. NEW SYNCHRONIZATION CONCEPTS

In its first section, this chapter focuses on the definition of the term *conflict* in order to follow up with a description of the new synchronization concepts. The second section introduces the *synchronization of complex objects* followed by the *dialogue-sensitive synchronization* in the third section.

2.1 Conflicts Defined

A conflict can only occur when two databases have a copy of a complex object CO with write-permission and the complex object is changed on both sides. A complex object consists of objects that conceptually belong together in the sense of real-world modelling. For more information on complex objects see [1].
CO_s complex object on the server; consists of data records from different tables $A \ldots Z$ called sub object $aA \ldots aZ$ with aA being the root sub object.
CO_1 Copy of CO on client 1.
CO_2 Copy of CO on client 2.
t_{d1}, t_{d2} point in time when CO_1, CO_2 are created/downloaded.
t_{c1}, t_{c2} point in time when sub object aI of CO_1 and sub object aJ of CO_2 are changed $\Rightarrow CO_1'$ and CO_2' with $t_{c1} \neq 0$ and $t_{c2} \neq 0$ (changes occurred on both sides).
t_{s1}, t_{s2} point in time when CO_1', CO_2' are synchronized.
t_{cs} point in time when complex object is changed on server.
Precondition:
$t_{d1} \leq t_{c1} \leq t_{s1}$ as well as $t_{d2} \leq t_{c2} \leq t_{s2}$

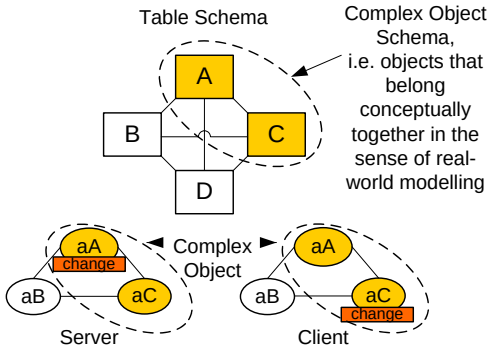

Figure 1: Synchronization Concept for Complex Objects

Assumption:
$t_{s1} < t_{s2}$ which means client 1 synchronizes before client 2
$\Rightarrow t_{cs} = t_{s1}$ (when client 1 synchronizes, its version is saved on the server).
A conflict will occur if:
$t_{d2} < t_{s1}$ (and complex object is changed on both clients (see precondition)).
Conflict will be detected at time t_{s2} when trying to set t_{cs} again without having seen the former version.

The detection of a conflict is possible by using *number ranges* for unique identification (see Chap. 5.3). Every client as well as the server has an own number range identifying all data records *uniquely*. As every sub object, the root sub object of a complex object has a unique ID. CO_1 and CO_2 are copies of the same complex object if the ID of the root sub object is identical.

The allowed operations are *Insert*, *Update* and *Delete*. Delete is only allowed in a few tables which are *checked out* (see Chap. 5.5) by the client. The conflict scenarios *Update-Delete* and *Delete-Delete* are therefore not possible. For the deletion of objects and the synchronization afterwards the concept of SyncServices (see Chap. 6.2) is used and a *tombstone table* maintains all deleted records. Through the synchronization process the entries in the tombstone table are sent from client to server, executed on server side and stored in the server tombstone table. When another client downloads data it will receive the new entries from the tombstone table and can delete the records locally.

Additionally, an *Insert-Insert* conflict is not possible either. Since every client and server has an own number range, the IDs will not be violated during synchronization.

The only possible conflicting operation is *Update-Update*.

2.2 Synchronization of Complex Objects

The *synchronization concept for complex objects* is shown in Fig. 1. For example, Table A may contain the zip code and city of an address. Table C contains the street name and street number. The objects conceptually belong together and form a complex object address.

Objects that conceptually belong together in the sense of real-world modelling are treated as one complex object so that conflicts can be detected even if data records from different data tables have been changed. A normal data replication approach would not detect the conflict and merge the data without displaying it to the user. Our approach offers a solution in this problem. Developers can specify which object classes form a complex object class. This information is then exploited at synchronization time.

There are three reasons why synchronizing complex objects can be a better approach:

- The *data model can be normalized* and does not have to consider dependent data. For example, an address can be split up in different tables with the street name in one table and the zip code in another table. If one user changes the street name of a person and another user changes the zip code, a merge can lead to a nonexistent address and is therefore undesirable. A conflict detection is necessary.
- The *data model can change*, nevertheless the same conflicts are detected. For example, the data model changes and splits up the person in two different tables. A data replication approach will detect two different conflict types based on the tables instead of one conflict for the entire person. When synchronizing complex objects the conflict will always be deteced within the entire person.
- *Detecting conflicts on objects* instead of data tables is more understandable for the user. For example, the complex object contact person has data from five different tables and is displayed to the user in one dialogue. If the user changes data on both, the client and the server he expects a conflict independent of the specific data records he has changed. In a data replication approach the user can change two different values and whether or not he will run into a conflict depends on the question if the two changed values are physically stored in the same table. Eventually, this might be confusing for the user.

Since *contact person* will be often used as an example for a complex object it will now be explained briefly. A *contact person* is a person being the contact person for a concrete company. One person can work for several companies and can therefore be contact person for more than one company. A contact person is a complex object containing of data from five different data tables. The data table *person* contains information about the person itself like name, title and birth date, the data table *contact person* provides information related to the company e.g. the e-mail address and telephone number of the person in the company, the data table *contact person function* contains the function of the contact person in the company e.g. CEO or secretary and the last data table is *address* which is connected to the person providing the private address and to the contact person containing the business address.

2.3 Dialogue-Sensitive Synchronization

The *dialogue-sensitive synchronization concept* is shown in Fig. 2. It offers a better decision support in the conflict resolution domain. The dialogue context where the change has occured is saved and presented to the user in the conflict display.

In Fig. 2 class A may stand for a contact person and class B may stand for a person. One physical person can be contact person for two different companies. The first user changes the person in the context of contact person A1 on the server whereas the second user changes the same person in the context of contact person A2 on the client. The dialogue-sensitive synchronization concept allows the user to see the conflict in the *context* of the contact person he has changed, so the server version of the conflict shows the person with contact person A1 and the client version shows the person with contact person A2.

In this paper, ”context” refers to the business logic view of objects rather than to the data view (e.g. person and contact person are two different tables but in business view they form one complex object).

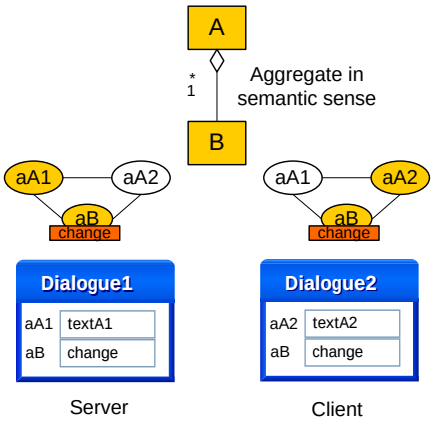

Figure 2: Dialogue-Sensitive Synchronization Concept

This concept is not limited to the same complex object. The first user can change a person in the context of the complex object patient, whereas the second user changes the same person in the context of a contact person.

3. RELATED WORK

Several papers have studied the issue of developing *synchronization for mobile environments* (e.g. [2, 3]). To improve synchronization, information about *data provenance* is needed, which is covered in detail in [4] and [5].

In [4] Foster and Karvounarakis studied provenance in the area of data replication for different devices. Replicas have to be transformed for different devices, which leads to difficulties in view update and maintenance.

In [5] Buneman et al. investigate data provenance and differ between "Why" and "Where" provenance. Data provenance describes where data comes from and the process in which the data was created or changed. "Why" refers to the source data that influenced the existence of the new data and "Where" identifies the origin location of the data.

In [6] Cui and Widom studied lineage tracing for data warehouses. During the integration of an operational data source into a data warehouse, source data is typically transformed. Data lineage covers the problem of tracing the derived data items to the original source items.

The process by which the data arrived in the database is also important in our approach in order to identify the complex object in which a data record was created or changed. Additionally, in a simple way this paper introduces a kind of "Who" provenance, where "Who" stands for the user who made the change in the database. This makes it possible to display the users, who made the changes that lead to a conflict. Similar to [4] the information is stored as metadata in the database in both cases. This is necessary to support the user with conflict resolution.

In [7] transaction processing techniques are described and used to monitor, control and update information. Transaction processing keeps a database in a consistent state by completing all transactions successfully or rolling them back otherwise. The book therefore covers fault tolerance, concurrency control as well as recovery and rollback. Keeping the database in a consistent state, identifying autonomous operations and rolling back transactions in case of an error is a demanding topic in most synchronization approaches.

In [8] Lee et al. studied data synchronization in mobile environment with focus on *conflict resolution*. For this implementation, SyncML was selected. Aspects of the synchronization are the usage of *Global Unique Identifiers* GUIDs to uniquely identify a data record and the usage of a *change log* which stores all changes that have to be synchronized. The advantage of a change log is a better performing synchronization since the single synchronization steps do not need to be reconfirmed and the execution process can be optimized locally. The aim of their approach is an *automatic synchronization* which needs a *conflict resolution policy* like originator-win, recipient-win, client-win, server-win, duplication or recent-data-win. From all these possible conflict resolution policies the recent-data-win policy has been selected. An automatic conflict resolution alleviates synchronization for the user and makes sense when all changes are made by the *same* user on different devices. However, when the changes are performed by *different* users, using a policy is a problem, especially when the policy decides that one change will overwrite the other. In contrast, our paper is based on the assumption of a mobile environment with many different users that change the same data in parallel, and it therefore offers an approach that supports manual conflict resolution.

Other related work about *replication of data for mobile environments* can be found in [9, 10]. In [9] Ratner et al. identified requirements for replication in a mobile environment. In [10] Barbara and Garcia-Molina studied dynamic replicated data management algorithms for generating and migrating replicated copies. Additionally, several different replication approaches for mobile systems are compared in [11]. Although a data replication approach might make things easier, there are some disadvantages to it, as well as requirements it can not fulfill. Chap. 4.1 will elaborate this fact.

4. SYNCHRONIZATION ARCHITECTURES

In this chapter two existing approaches for exchanging data between client and server are illustrated and evaluated. The first approach is the *Data-Centric Architecture* followed by the *Service-Oriented Architecture*. Since both approaches do not fit the requirements, a new architecture approach, the *Context-Oriented Architecture* has been developed and is discussed in Chap. 4.3.

4.1 Data-Centric Architecture

In a *Data-Centric Architecture* shown in Fig. 3 the database of the server is *fully or partly replicated* to the client database and data differences are merged. Changes can be tracked and conflicts can be detected.

Oracle supports synchronization between two Oracle databases with conflict detection [12]. However, the conflict resolution is based on simple rules like client-wins, server-wins or a custom programmatic resolution.

The advantages of this approach are that a lot of research has been made in this area and that well tested solutions are available. The disadvantages are that for this approach both databases must be *compatible*.

Additionally, the server database must be *reached directly* by the client. This can be acceptable when the server database is only exposed to a secure intranet but might be a security risk if the clients access from the Internet as required in the approach described in this paper. Moreover, a conflict resolution policy will *automatically override changes* of one user. Even if there would be the possibility to display the conflict and let the user decide, a conflict would only be displayed based on the *actual table* but not on the *context* in which the object was changed. See Chap. 5.6 for more details on context-oriented conflict detection and resolution. Another disadvantage of Data-Centric Architecture is the difficulty of *replicating*

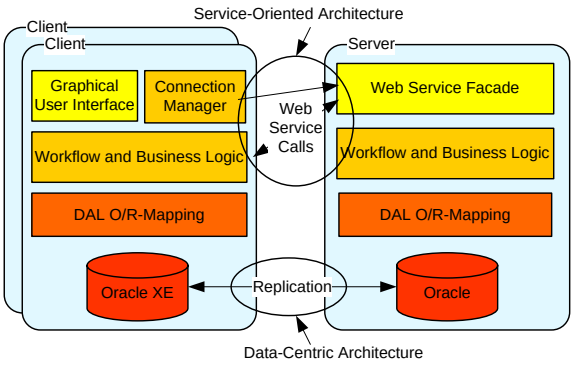

Figure 3: Data-Centric vs. Service-Oriented Architecture

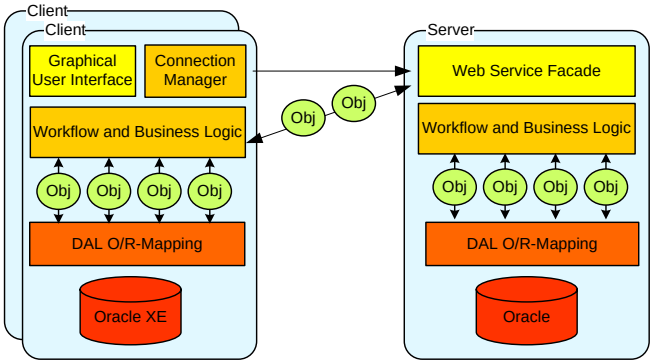

Figure 4: Context-Oriented Architecture

parts of a table based on objects as discussed for the SyncServices in Chap. 6.2. The required where-statements will be quite complex. Last but not least it is sometimes necessary to change data during replication, e.g. check data out and save it without a lock on the client or load data and save it read-only on the client. After considering these issues, a data-centric approach was not the right choice to fulfill all requirements.

4.2 Service-Oriented Architecture

The goal of *Service-Oriented Architecture* (SOA) (also shown in Fig. 3) is *loose coupling* between interacting components. Therefore, a service, which can be a *Web Service*, is offered on the net. The business logic on the client calls the server-sided Web Service. The data model of client and server can be different and in addition, the client is responsible for conflict detection and resolution, which requires custom-implemented conflict handling. Advantages of SOA are the *independence of the database*, so databases from different vendors can be used, as well as the possibility to provide better security mechanism for the server database. A disadvantage of SOA compared to Data-Centric Architecture is *worse performance*. However, the concept of synchronization over Web Services is also used in the Context-Oriented Synchronization Approach.

4.3 Context-Oriented Architecture

As both described architectures are not capable of detecting and resolving a conflict based on the context, a new approach, the *Context-Oriented Architecture*, is developed and shown in Fig. 4. Similar to the *Service-Oriented Architecture*, this approach uses a *Web Service* for communication with the server. The synchronization however is based on *objects* which are created using an O/R-Mapping tool. Every object has data from the database which is not only composed of a single relational record but several records across many tables. Client and server have relational databases with an additional table to maintain the metadata of the objects and save the context of changes.

Compared to the *Data-Centric Architecture*, the data model on client and server has to be similar, to be able to create compatible objects. However, a Data-Centric Architecture disposes of only basic conflict handling capabilities and does not support business processes during synchronization. As an example in PreVolution it is a requirement, that an offline created company has to be forwarded to the supervisor for approval during synchronization.

The Context-Oriented Architecture is similar to the Service-Oriented Architecture as both use Web Services for communication and benefit from loose coupling. Still, Service-Oriented Architecture is defined as requiring independent and committed transactions, which is not the case in our approach. The synchronization requires several dependent Web Service calls.

The advantage of the Context-Oriented Architecture is that as it is based on objects, it is also possible to *synchronize defined subsets* of the database and still guarantee *consistency*. This is an important feature as illustrated in the following example: Some tables on the server-sided database are insert-once restricted and data from these tables can be kept offline and changed several times until the user releases and commits the data. Nevertheless it is possible to synchronize the rest of the data in between, providing better data safety and performance through partial upload.

5. CONTEXT-ORIENTED SYNCHRONIZATION

This chapter describes the *implementation* of the *Context-Oriented Synchronization Approach* in PreVolution in detail. The first section gives an overview of the *architecture*, the second briefly describes the *Genome* O/R-Mapping tool and the third section explains *data model* and *number ranges*. The fourth section deals with data *upload* from client to server, whereas the fifth section discusses the opposite direction, *downloading* data from server to client. The sixth section covers the demanding topic of *conflict detection and resolution* and the last section elaborates on some *challenges* encountered while implementing the Context-Oriented Synchronization Approach.

5.1 Architecture

This section describes the *architecture* of PreVolution, as displayed in Fig. 4. On the client side *Oracle XE* is used as local database, while *Oracle Database 10g* is used on the server side.

In a previous implementation *Microsoft SQL Desktop Engine* (MSDE) was used on the client, but incompatibilities between the two databases were encountered (e.g. Unique in SQL allows only one NULL value, whereas Oracle can have several NULL values, or slightly different data types between the two databases) so MSDE was exchanged for Oracle XE. However our new synchronization approach does not need two databases from the same vendor, it works with MSDE as well.

Both, client and server, use *Genome* (see Chap. 5.2) as Data Access Layer and O/R-Mapping tool.

The *Business Logic* (shown in Fig. 5) is identical on client and server, so there is no difference between working online or offline. The client uses the ILogic Interface and depending on the client being online or offline, the specific instance of ILogic is either LogicToServer or LogicToDatabase. When being online, LogicToServer

is called, the call is forwarded to the Web Service Façade and after calling a Controller the logic in LogicToDatabase is executed, which queries the server-sided Oracle database. When the client is offline, LogicToDatabase is called directly and queries are executed on the local Oracle XE instance.

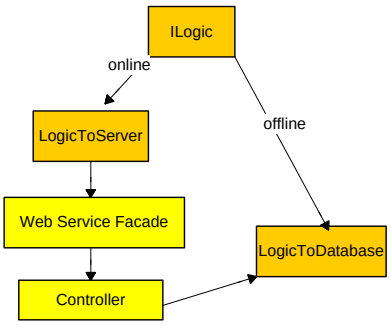

Figure 5: Business Logic

Finally the client provides the graphical user interface and the connection manager from the Smart Client Offline Application Block (see Chap. 6.1), whereas the server includes a Web Service Façade.

5.2 O/R-Mapping with Genome

Genome [13] developed by *TechTalk* is used as *Data Access Layer*. Genome is an O/R-Mapping tool for .NET and supports Oracle, Microsoft SQL Server and IBM DB2 as databases. Genome supports lazy loading (objects of a result set or attributes of an object are not loaded until they are needed) as well as *optimistic* and *pessimistic locking*.

Optimistic locking allows concurrent access, however when several clients try to commit changes, only the first client commits successfully and the other clients receive an exception. The disadvantage of optimistic locking in Genome is that an additional *version field* is necessary in every database table. As this version field has to be set by every application updating a row in the database, optimistic locking was not possible in the approach described in this paper since other applications use the same database.

The advantage of *pessimistic locking* is that the client knows upfront that data is locked and that it will not lose any changed data. In the present approach pessimistic locking is used and data will only be committed once the lock is released. Additionally the communication with the database is executed in a context which differs between *ReadOnlyContext* for read-only and *ShortRunningTransactionContext* for read/write.

The description of the data model in Genome is XML-based. For each table an Entity is created and these Entities are combined to Objects. There are two different kinds of objects: the first are the normal Genome-Objects, which are high-performance, but can not be transmitted over a Web Service. The second type is the Data Transfer Objects (DTOs), which are defined through Genome Views and can be transmitted over a Web Service. A disadvantage of Genome is that it only offers methods to serialize the Entities based on a View into a DTO but not to deserialize them on the other side.

Supported query languages are Object Query Language (OQL) and LINQ, which is new and was not supported until October 2007. Although Genome is an important and very beneficial tool in PreVolution, it has no mechanism for either *synchronization* or *conflict handling*.

5.3 Data Model and Identification

As common in many projects, the database on the server is not only used by PreVolution but also by other existing applications inside the AUVA. Therefore changes in the existing data model are difficult. *Global Unique Identifiers* (GUID) as used in [8] would have been the ideal method for data record identification. However, every table has a 64-bit long number as a primary key. Another approach was using *negative IDs* on the client for new records and replacing them later during uploading, but this is complex and costly and conflicts would have been possible on the server with new records from other applications. The best-fitting approach was using *number ranges* on server and client side. A new client without a number range, requests the number range from the server and all created records on the client receive an ID from this range. All used ranges are maintained in a database table on the server.

Every table on client and server has eight *additional columns* containing user and date information in columns *CreatedBy*, *DateCreated*, *ModifiedBy*, *DateModified*, *LockBy*, *LockDate*, and *LockType* for recognizing changes and *SyncDate* for the synchronization. The three primary columns are *DateModified*, *LockType* and *SyncDate*. Whenever a change occurs *DateModified* is set to the current date. *LockType* allows checking out data from the server or save read-only data on the client. Every lock is saved in this column. Very important for the synchronization is the client-sided *SyncDate* column, which is not populated on server-side. It contains the date at which the record was last loaded from the server. Microsoft SyncServices (see Chap. 6.2) have less but similar columns for maintaining changes.

Another aspect is the *SyncTable*. This table enables Context-Oriented Synchronization and allows context-based conflict resolution by saving all objects. Besides the *ID* this table provides the columns *ObjectType*, *ObjectReference*, *conflicts* (bool), *SyncCreatedDate*, *SyncModifiedDate* and *LastSyncDate*. When downloading a contact person with trailed records from the tables *person*, *address* and *contact person functions*, one entry would be inserted in the *SyncTable*. *ObjectType* would be contact person, *ObjectReference* the ID of the contact person in the contact person table and *conflict* would be false. *LastSyncDate* and *SyncModifiedDate* would be the same date, which has been transmitted from the server. When the client is now offline and changes a property in the table *contact person function*, the actual date is saved in *DateModified* of the *contact person function* table and additionally in the *SyncModifiedDate* of the *SyncTable* for the concrete complex object contact person. When synchronizing the changed data can be extracted from the *SyncTable* by comparing *SyncModifiedDate* with *LastSyncDate*, and if a conflict occurs in the function, the conflict can be resolved within the context of the contact person. For more information on conflicts see Chap. 5.6.

The next change in the data model is a server-side table called *ReplicationTable*.When the client starts synchronization, the ID and the name of every complex object (e.g. person) that has to be transmitted to the client is stored in this table. The first advantage is that packages of e.g. 100 persons can be requested by the client and stored in the local database. It is not necessary to send all data at once. The second advantage is that the client can abort the synchronization process after each package and, for example, continue the next day. In this case the *ReplicationTable* is updated with data that changed in the meantime, and the client can continue the synchronization process.

The data models on client and server are largely the same, although some tables are only used by the client or by the server. However there is one *additional role* on the client which has the right to delete records from all tables. No role has the right to

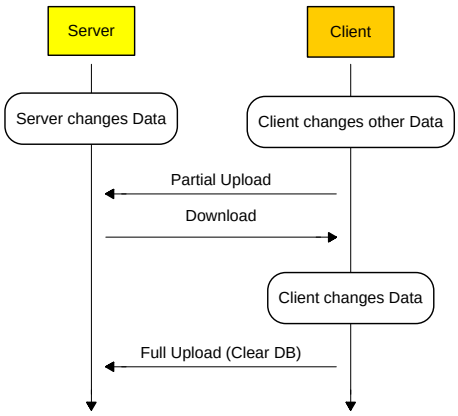

Figure 6: Synchronization Flow

delete records on the server.

5.4 Upload

As already mentioned before, *Uploading* in this paper characterizes the upload of client data to the server. There are two different types of uploads, namely *full upload* and *partial upload* (see Fig. 6). Both identify all offline changed data and send them to the server. If conflicts occur, both will *detect the conflicts* and ask the user to resolve them. However, the partial upload checks if there are offline changed data available before starting the upload, and locks on the server will not be removed. The full upload does not check for changes, it will always start the upload and if nothing has changed, at least the locks will be removed on the server.

The *partial upload* is used *implicitly* when the user wants to download data. The implicit upload before downloading is necessary, because only the upload can detect conflicts.

The *full upload* is started *explicitly* by the user and after sending all changed data to the server and resolving potential conflicts, the local database will be largely cleared, except of a few tables, which need to be always available offline and are therefore never cleared.

There is one more aspect that needs to be mentioned, because it is one of the challenges described in Chap. 5.7. Addresses are standardized and allowed to exist only once in the database. When the user creates offline a new address for a company and this address already exists on the server, during upload the objects loaded from the client are changed to point to the existing address on the server. The existing address is sent back to the client and stored in the database. All objects are changed to point to the existing address from the server and afterwards the client-sided address is deleted.

5.5 Download

Downloading refers to the download of data from server to client. A download can only occur after a partial upload, so it is not possible to run into conflicts during downloads.

There are three different *types* of data to be downloaded:

- The first group is data that are downloaded *read-only*. The existing workflow as well as old orders and documents belong to this group.
- The second group consists of data that is *checked out* from the server, so the server gets a lock and the data can only be changed on the client. For example, the current order and the route belong to this group. No other user should change the same order at the same time. In these two groups conflicts are not possible, because data are only changed on one side.
- However, there is a third group which includes data that can be *changed on both sides*. Contact persons, patients or economic domains belong to this group.

If the user downloads again, the client-sided *LastSyncDate* is compared with the server-sided *ModifiedDate* and changed data is downloaded again.

If the user has not yet downloaded or performed a full upload before, and the connection breaks, he can still work offline. There are data, which are always available offline, and these data are enough for working with the application.

5.6 Conflict Detection and Resolution

As already mentioned, conflicts are only detected during upload. The client sends its changed data and the *LastSyncDate* to the server, and if the server has changed the same data, a conflict is detected.

Conflicts are detected and displayed *context-based*, this means that if the client changes the address of a person from Main Street 3a to Main Street 3b and another user changes the address of the same person to Main Street 3c, a conflict resolution without context would display Main Street 3b versus Main Street 3c. The user would not be able to decide, because he would not know why and for whom the address was changed. It would not be possible to determine the person, for which the address was changed, when several objects like other persons and companies point to the address too. However, the Context-Oriented Synchronization knows in which context the address was changed and is able to detect and display the conflict for the person with the address.

There are different possibilities to define a conflict. The first kind of conflict is when an *attribute* is changed on both sides and there is no possibility to merge. This would be the case in the address example above, because 3b and 3c cannot be merged. Another type of conflict is characterized by being based on a *complex object* instead of a single attribute. For example, the client can change the e-mail address of a contact person and a user on the server changes the name of the person (note that contact person and person are two different tables). Without context-oriented conflict detection it would not be possible to detect this conflict during upload, because when uploading the persons, no conflict would occur. When uploading the contact persons later, there would be no conflict either. In our approach it is possible to detect this conflict because the *SyncTable* tracks that something has changed in the context of a contact person and uploads the whole contact person as one complex object.

All conflicts are shown to the user when he is online. He sees the server version with the name of the user who changed it and his version with the later changed preselected and can decide which one to take. The server version of conflicts are regularly updated, since it is possible that another user changes the server version again. When the user resolves a conflict, it is again checked whether the server version has changed in the meantime. If it has, there is a *conflict on a conflict* and the user has to decide again which version to take.

5.7 Challenges

The first challenge in this project was the server-sided data model, which was not allowed to be changed. *Number ranges* have to be used instead of *Global Unique Identifiers* and the additional columns for *change tracking* had to be inserted into separate tables. A view then combined the two tables. As deletion on server side is

only allowed for a few tables, there have been some cases, where the ID from the server has to be taken and replaced on the client during upload e.g. for standardized addresses.

Another challenge are *unique constraints* which are not directly part of the synchronization process but problems can occur during upload. The synchronization will not treat this as a conflict but it has to know all unique constraints and if during upload one constraint is violated, the user has to be informed that he has to correct the data offline before trying to upload again.

Error handling and *recovery* was another challenge. When an exception occurs during synchronization it is important that both databases remain in a consistent state. This is achieved on the one hand by using the Genome *transaction context* as described in Chap. 5.2. Using this approach allowed using Genome's generic transaction manager instead of using the specific transaction manager provided by an underlying database. A *ShortRunningTransactionContext* is opened and all changes are defined until one commit persists all changes at once into the database. Therefore partly committed data within one context will never occur.

On the other hand all methods are designed such that they are *independent* from each other. With the help of the *ReplicationTable* described in Chap. 5.3 it is possible to download independent packages of person objects. After saving a package in the local database the client sends a *delivery receipt* for the package. If the package is already stored locally and an exception occurs when sending the receipt, the server will resend the same package. If the client does not get a *delivery receipt* from the server after uploading, it will also resend the data.

As a result of an exception it is possible that the data in the local database has *different synchronization dates*, however this is also possible when the user stops the synchronization manually and does not limit the use of the application.

The last challenge and still not resolved at the moment is the *complete rollback* of the synchronization process. One requirement demands that a rollback is always possible throughout the entire synchronization process. However, some data are stored on the server and some on the client and there are several Web Service calls in between. As the Genome transaction can not be kept open for such a long duration, this is not easily solvable. However, there are other options, for example to allow a rollback after each step during upload and download as described above so that the data are always consistent.

6. SYNCHRONIZATION TECHNOLOGIES

After describing the *Context-Oriented Synchronizaion Approach* and a possible implementation, this chapter will briefly introduce *two promising technologies* from Microsoft. The first is the *Smart Client Offline Application Block* [14] followed by the *Sync Services for ADO.NET* [15]. Both technologies provided ideas for and allow comparison with the new approach.

6.1 Smart Client Offline Application Block

The *Smart Client Offline Application Block* (SCOAB) [14] is an Application Block from Microsoft that has been published first in 2004, including best practices for the design of an architecture and for solving problems in the context of online/offline scenarios. SCOAB comprises *best practices*, *design patterns* and *examples*. Additionally, the entire code is available as well, allowing to adapt the Application Block for individual needs. SCOAB is meant to support the development of offline-capable applications. Therefore, the data are stored on the client and SCOAB takes care of the synchronization as soon as a connection is available.

SCOAB uses *DataSets* for conflict detection. When implementing a prototype in 2005, DataSets did not support conflict resolution, but basic abilities were added in the meantime. For communication with the server a Web Service is used and messages are sent through *Microsoft Message Queuing* (MSMQ). For data management *In-Memory*, *Microsoft Desktop Engine* (MSDE) or *Microsoft SQL Server* can be used. SCOAB uses several threads, one for the application, a second for the connection manager and a third for the messages. Additionally, SCOAB includes an extensive evaluation of possible security risks, as well as suggestions for testcases. SCOAB is based on .NET-Framework 1.1 but can be upgraded to .NET-Framework 2.0.

A prototype at the beginning of PreVolution was developed to evaluate SCOAB. However, SCOAB was largely not included in the final version of the project since DataSets were not sufficient. One component of SCOAB, the *connection manager*, which detects if a connection to the server is available and allows switching between online and offline modes, has been integrated into PreVolution in slightly modified manner.

6.2 Synchronization Services for ADO.NET

The *SyncServices for ADO.NET* [15] are a completely new part of the *Microsoft Sync Framework* (MSF) and *.NET Framework 3.5*, which is integrated in Visual Studio 2008. SyncServices allow four different synchronization methods:

- Snapshot
- Incremental Download (Insert, Update, Delete)
- Upload (Insert, Update, Delete)
- Bidirectional Synchronization with Conflict Handling

Microsoft SQL Server Compact Edition 3.5 (CE) has to be used as local database, which is only slightly different from *Microsoft SQL Express Edition*, but takes less space, is stored in one file and can store data up to 4 GB. The database on the server can be *Microsoft SQL Server* or another database like *Oracle* or *DB2*. SyncServices allow communication over Web Services. SyncTable objects determine the tables to be synchronized; several SyncTables can be grouped into a SyncGroup, which is always synchronized within one transaction, so a rollback is possible.

SyncServices were evaluated for approximately one month in several prototypes and demos but were not appropriate for our requirements due to the following issues:

- *Conflict resolution is not fully developed* at the moment, some conflicts have to be resolved on the server or the process has to be suspended.
- SyncServices are *completely new*, Visual Studio 2008 has to be used and documentation is rare. Additionally, Microsoft SQL Server Compact 3.5, which has some compatibility problems with Oracle, is a must on the client.
- *Changes while downloading* (e.g. setting a lock) cannot be done during the synchronization process but require a separate SQL-Statement. However, this would be detected as a change and propagated to the server during the next synchronization process.
- *Synchronization of objects is not possible*, instead tables are synchronized. The records in a table can be restricted using Select-Statements. These statements would be very extensive, because there is no need to download all two million

addresses but only those connected to a person, a contact person, a company or other objects the user selects in his download list, which likely will result in approximately 600.000 addresses. The list of IDs in the Select-Statement will nevertheless be quite large.

- *Restriction of the data amount* is only possible on the server but not on the client. A partial upload from the client to the server which offers better performance on the one hand, and allows to synchronize data with insert-once restriction on the server on the other hand, is not possible with one Sync Agent. However, using different Sync Agents leads to other problems including rollbacks. Normally, rollback of transactions is supported in an efficient manner, but not possible when using different Sync Agents.
- The last issue is that the *code is not available*, as it would be in SCOAB or a custom-implemented approach. Therefore, the code cannot be extended and necessary changes cannot be implemented.

7. LESSONS LEARNED

In this project lessons were learned regarding workflow management [16], requirements engineering [17] as well as regarding synchronization. There is a huge number of different approaches for synchronization and it is difficult to decide. Evaluation takes some time and sooner or later a decision has to be made. SCOAB and SyncServices were the main but not the only evaluated technologies, however no technology fitted and an *own approach* had to be implemented. Building an own synchronization solution is not as easy as it first seems. Synchronization is something that should work in the *background* and not bother the user until it is really necessary; nevertheless, synchronization is a *main component* of every online/offline application and is quite complex. Every possible scenario can and will happen, e.g. conflict during conflict resolution. However, implementing an own synchronization approach is possible.

Another important lesson is that also if a synchronization approach does not necessarily need two databases from the same vendor or a similar data model, it is always an advantage to do so. If it is possible to have data in the *same structure*, schema mapping can be avoided, thereby reducing complexity and effort.

The last important lesson learned is that a local database has *limits* compared to the server database. Knowing that an offline database is available may tempt to think that it would also be nice to have more and more data offline. However, an offline database has a size limit and besides the synchronization process should be kept simple and fast, so it is highly advisable to download only what is really necessary and helpful, and if some data are only needed for information porposes, they should be made read-only in order to reduce the possibility of conflicts.

8. ACKNOWLEDGEMENTS

The authors gratefully acknowledge support by the Austrian Government, the State of Upper Austria, and the Johannes Kepler University Linz in the framework of the Kplus Competence Center Program. Special thanks go to all project members of PreVolution, working and have worked in this project.

9. REFERENCES

[1] Dirk Draheim and Gerald Weber: Form-Oriented Analysis: A New Methodology to Model Form-Based Applications, Springer Verlag, 2005.

[2] Shirish Hemant Phatak and B. R. Badrinath: Conflict Resolution and Reconciliation in Disconnected Databases. 10th International Workshop on Database and Expert Systems Applications (DEXA), 1999.

[3] Alan Demers, Karin Petersen, Mike Spreitzer, Douglas Terry, Marvin Theimer, and Brent Welch: The Bayou Architecture: Support for Data Sharing among Mobile Users. Proceedings IEEE Workshop on Mobile Computing Systems & Applications, 1994.

[4] J. Nathan Foster and Grigoris Karvounarakis. Provenance and Data Synchronization. IEEE Data Engineering Bulletin, Volume 30, 2007.

[5] Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and where: A Characterization of Data Provenance. In International Conference on Database Theory (ICDT), 2001.

[6] Yingwei Cui and Jennifer Widom: Lineage Tracing for General Data Warehouse Transformations. Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01). 2001.

[7] Jim Gray and Andreas Reuter: Transaction Processing: Concepts and Techniques, Morgan Kaufmann, 1993.

[8] YoungSeok Lee, YounSoo Kim, and Hoon Choi: Conflict Resolution of Data Synchronization in Mobile Environment. Computational Science and Its Applications - ICCSA, 2004.

[9] David Ratner, Peter Reiher, Gerald J. Popek, and Geoffrey H. Kuenning. Replication requirements in mobile environments. Mobile Networks and Applications, Volume 6, 2001.

[10] Daniel Barbara and Hector Garcia-Molina. Replicated Data Management in Mobile Environments: Anything New Under the Sun? Applications in Parallel and Distributed Computing, 1994.

[11] Astrid Lubinski and Andreas Heuer: Configured replication for mobile applications. Proceedings of the IEEE International Baltic Workshop on DB and IS, BalticDB&IS'2000, 2000.

[12] Philip Stephenson. Oracle Database Lite 10g Technical White Paper, May 2005.

[13] TechTalk. Genome - Generative Object Mapping Engine, White Paper, 2006.

[14] Microsoft. Smart Client Offline Application Block. URL, http://msdn2.microsoft.com/en-us/library/ms998460.aspx, 2004.

[15] Microsoft. Sync Services for ADO.NET. URL, http://msdn2.microsoft.com/en-us/sync/bb887608.aspx, 2008.

[16] Theodorich Kopetzky and Dirk Draheim. Workflow Management and Service Oriented Architecture. SEKE 2007, pages 749-750, 2007.

[17] Mario Pichler, Hildegard Rumetshofer, and Wilhelm Wahler. Agile requirements engineering for a social insurance for occupational risks organization: A case study. In RE '06: Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06), pages 246-251, Washington, DC, USA, 2006. IEEE Computer Society.

Optimization of Preference Queries with Multiple Constraints

Markus Endres and Werner Kießling
University of Augsburg
Institute for Computer Science
Germany
{endres,kiessling}@informatik.uni-augsburg.de

ABSTRACT

Nowadays, the efficient integration of preference querying into standard database technology is an important issue. In some instances, preference queries challenge traditional query processing and optimization. In this paper we study preference database queries involving hard constraints over multiple attributes belonging to several relations. The main bottleneck for such queries is the computation of the cartesian product which may lead to high memory and computation costs. We develop algebraic optimization techniques to transform a preference query with hard constraints in order to enable its efficient processing by database engines. For this purpose, we show a dominance criterion and we introduce rewriting techniques to eliminate dominated tuples before building the cartesian product and therefore speed up the evaluation. These techniques lead to novel preference transformation laws and extend previous developed rules.

1. INTRODUCTION

Preferences are ubiquitous in everyday private and business life. They recently have received increased attention in the scope of personalized applications. In many cases they are designed for use in database search engines and e-commerce applications (e.g. [4, 7, 16]). In some instances, preference queries challenge traditional query processing and optimization, as illustrated by the following example.

EXAMPLE 1. *Consider a database storing nutritional information for single servings of different kinds of food relations like Soups, Meats and Beverages. A user, Mrs. Diet, wants to complete her diet sheet and therefore is interested in finding meals that satisfy nutritional requirements such as a restriction on the number of calories (cal), the amount of Vitamin C (Vc) and the amount of total lipid fat (abbr. fat). For example, the recommendations for a 30-year old female, who is moderately active, are at most 1100 calories, at least 38mg of Vitamin C and maximal 9g of fat for a main meal [24].*

However, in the context of a diet, each user has preferences concerning its meals. Mrs. Diet for example ***likes chicken soup*** *as starter. The main course* ***should be beef*** *and the* ***cholesterol*** *of the beef* ***should be as little as possible****. For beverage she* ***likes red wine****. These preferences are equally important for her (a Pareto preference). The complete meal* ***must fulfill*** *the restrictions of maximal* ***1100 calories****, at least* ***38g of vitamin C*** *and at most* ***9g fat*** *for her personalized diet sheet.*

Mrs. Diet wants find all such meals which fulfill the hard constraints and satisfy her preferences best possible.

Using Kießling's approach of modelling preferences as strict partial orders [13, 14], the above mentioned hard and soft constraints can be expressed by Preference SQL [16] as follows:

```
SELECT S.name, M.name, B.name
FROM   Soups S, Meats M, Beverages B
WHERE  S.cal + M.cal + B.cal ≤ 1100
  AND  S.Vc + M.Vc + B.Vc ≥ 38
  AND  S.fat + M.fat + B.fat ≤ 9
PREFERRING
  S.name IN ('Chicken soup')        AND
 (M.name IN ('Beef')
          AND M.Cholesterol LOWEST) AND
  B.name IN ('Red wine')
```

This query expresses Mrs. Diet's preferences after the keyword `PREFERRING`*. It is a Pareto preference (*`AND`*)* [1] *consisting of preferences on soups, meats and beverages.* `IN` *denotes a preference for members of a given set, a POS-preference. The whole preference is evaluated on the result of the hard sum constraints.*

The query in the example is a preference query containing multiple sum constraints in its condition and user preferences on some attributes. Therefore, we call such queries **Preference Queries with Multiple Constraints**.

Of course, those problems are not limited to planning tasks as in the example above. They also occur in the context of document retrieval [12, 8], multimedia data retrieval [6], geographic information systems [9], dynamic resource allocation on the grid [18] and e-commerce [22, 4].

Conventional approaches implement such queries by a set of binary join operators and evaluate the hard constraints. Afterwards the user preferences as soft selection combined with the Pareto operator (`AND`, $\otimes$) are evaluated by a skyline algorithm, e.g. [2, 21, 20], to retrieve all combinations that fulfill the preferences best possible.

Because the hard constraints refer to attributes from more than two relations, pair-wise join operators may fail to remove intermediate results based on these condition. For a hard constraint such

[1]Note that `AND` in the `WHERE`-clause means Boolean conjunction, whereas `AND` in the `PREFERRING`-clause denotes Pareto preference construction, i.e. all preferences are equally important.

VLDB '08, August 24-30, 2008, Auckland, New Zealand

as $A_1 + ... + A_r \ \Theta \ c, \ \Theta \in \{<, >, \leq, \geq, =, \neq\}$ and c a constant, current join operators cannot test the satisfiability of an intermediate tuple until all variables have been determined. A consequence of this inability to remove intermediate tuples that will not lead to any results is that the query evaluation process must evaluate the cartesian product of all tuples of all join relations, which leads to **high memory and computation costs**, particularly if the relations are large.

The goal of this paper is to **eliminate tuples** from the relations which definitely can never be in the result set **before building the cartesian product**. This reduces the relation sizes and therefore the computation costs and needed memory for the cartesian product.

Section 4.1 of this paper is an extension of [5], where we only considered **one single sum constraint** over multiple attributes. But in real life, database queries are not restricted to only one hard constraint, but use several different constraints, e.g. as in example 1. Therefore, we must consider **multiple hard constraints** over multiple attributes and for such problems we present novel rewriting techniques in this paper. Further on, we present optimization techniques based on mathematical rewriting of the original constraints and add additional **selection operators** into the Preference SQL query, cp. section 4.2.

The rest of this paper is organized as follows: In the next section we take a look at related work. Afterwards we describe the background of our preference model in section 3. In section 4 we develop different optimization techniques and introduce new rewriting techniques for preference queries. Further on, we present experimental results in section 5, and finally we conclude with a summary and outlook.

2. RELATED WORK

Database queries with constraints over attributes belonging to several relations occur frequently in the real world, but have not been intensively researched in the database context in the past.

Agarwal et. al. [1] address queries with linear constraints, and Guha et. al. [10] address queries with aggregation constraints. However, their work is only valid for queries on one relation.

Ilyas et. al. [12] developed algorithms for top-k queries that can be extended to implement queries with a constraint on the value of a monotone function, but this is only valid for one constraint. Liu, Yang and Foster [17] integrated constraint-programming techniques with traditional database techniques to solve sum constraint queries by modifiying existing nested loop-join operators. Nestorov, Liu and Foster [19] use similar principles but can be implemented without modifying existing database engines, but not in combination with preferences.

Georgiadis et. al. [8] prevent the construction of tuples that cannot appear in the result of a preference query using appropriate linearizations, but not in combination with hard constraints.

Hafenrichter [11] and Chomicki [3] developed extensive transformation laws and rewriting techniques for various preference queries, e.g. 'push preference over hard constraint', but these laws are not applicable for the processing of preference queries with hard constraints over multiple attributes belonging to more than one relation.

Döring et. al. [4] present a first approach for processing of queries dealing with individual and global preferences of customers. Their preference query rewriting is a kind of query expansion to deliver all tuples that matches the original preferences of a user as well as cheaper alternatives in an online travel booking system.

In [5] we developed transformation laws to optimize preference queries with **one sum constraint** over a set of attributes belonging to several relations. But these laws are only valid for one constraint. In this paper we develop laws for **multiple constraints** and introduce further optimization techniques. Further on, we show how to integrate our new laws into the preference query optimizer implemented by Hafenrichter [11].

Note that our approach does not intend to replace other optimization methods. Instead, our method can be combined with other algorithms to allow more *Optimization of Preference Queries with Multiple Constraints* on relational database systems.

3. PREFERENCE ALGEBRA

In this section we want to give some background concerning the preference technology.

3.1 Preference Algebra

Preference frameworks tailored to standard database systems have been introduced in [13] and [3]. We depict the preference algebra from [13] which is a direct mapping to relational algebra and declarative query languages. This preference model is based on *strict partial orders* and is semantically rich, easy to handle and very flexible to represent user preferences which are ubiquitous in our life.

DEFINITION 1. ***Preference***
Let $A = \{A_1, \ldots, A_n\}$ be a set of attribute names with corresponding domains of values $dom(A_i)$. The domain of A is defined as $dom(A) = dom(A_1) \times \cdots \times dom(A_n)$. Then a preference P is a strict partial order $P = (A, <_P)$, where $<_P \subseteq \ dom(A) \times dom(A)$.
The term $x <_P y$ is interpreted as "I like y more than x".

Having defined preferences as strict partial orders we provide a variety of intuitive and customizable **base preference constructors** for categorical and numerical domains which can intuitively be combined to build complex preferences still yielding partial orders. Formally, a base preference constructor has one or more arguments, the first characterizing the attributes A and the others the strict partial order $<_P$, referring to A.

For example, the POS-preference $POS(A, <_P)$ on an attribute A expresses that a special value of an attribute is preferred to all others (compare the `IN`-clause in example 1). There is also a NEG preference constructor, which is defined analogous to the POS-preference, but with a NEG-set. Moreover, it is possible to combine these preferences to *POS/NEG* or *POS/POS*.

If we want to focus on preferences where the domain is a numerical data type, e.g. decimal, which can be infinite, we can use a number of *numerical preference constructors* which are defined by [13], e.g. LOWEST, HIGHEST, AROUND, BETWEEN and the SCORE preference. LOWEST(cholesterol), for example, represents that a person prefers lower values for 'cholesterol' over higher values.

DEFINITION 2. ***Extremal preferences***
We define the LOWEST and HIGHEST preference that the desired value should be as low (high) as possible. Formally:

- *P is called a LOWEST preference, if: $x <_P y$ iff $x > y$*
- *P is called a HIGHEST preference, if: $x <_P y$ iff $x < y$*

There are further categorical and numerical base preference constructors. Detailed information on these preference constructors are given in [13].

In the following we will briefly discuss two **complex preference constructors**, namely the *Pareto preference* (`AND`, $\otimes$) and the *Prioritization* (`PRIOR TO, &`).

DEFINITION 3. ***Pareto preference:*** $P_1 \otimes P_2$
Given two preferences $P_1 = (A, <_{P_1})$ *and* $P_2 = (B, <_{P_2})$*, for* $x, y \in dom(A) \times dom(B)$ *we define*

$$x <_{P_1 \otimes P_2} y \quad \textit{iff}$$
$$(x_1 <_{P_1} y_1 \wedge (x_2 <_{P_2} y_2 \vee x_2 = y_2)) \quad \vee$$
$$(x_2 <_{P_2} y_2 \wedge (x_1 <_{P_1} y_1 \vee x_1 = y_1))$$

$P = (A \cup B, <_{P_1 \otimes P_2})$ *is called Pareto preference modelling* P_1 *and* P_2 *as* ***equally important****.*

DEFINITION 4. ***Prioritized preference:*** $P_1 \,\&\, P_2$
Given two preferences $P_1 = (A, <_{P_1})$ *and* $P_2 = (B, <_{P_2})$*, for* $x, y \in dom(A) \times dom(B)$ *we define*

$$x <_{P_1 \& P_2} y \quad \textit{iff}$$
$$x_1 <_{P_1} y_1 \vee (x_1 = y_1 \wedge x_2 <_{P_2} y_2)$$

$P1$ *is considered* ***more important*** *than* P_2*;* P_2 *is respected only where* P_1 *does not mind.*

A generalization of the Pareto preference constructor and the Prioritization to more than two preferences is obvious. An extended definition can be found in [21, 20]. Detailed information on all preference constructors are given in [13].

EXAMPLE 2. *As in example 1, beef and cholesterol as little as possible are equally important for Mrs. Diet. With preference algebra we describe her preferences as:*

$$P = POS(Meats, \{Beef\}) \otimes LOWEST(cholesterol)$$

3.2 The BMO Query Model

Given preferences over a set of attributes a central question is to determine an outcome that is preferentially optimal with respect to the preference statements. Whether preferences can be satisfied depends on the current database contents. Thus a match-making between wishes and data has to be made.

For this purpose the **Best-Matches-Only (BMO)** query model has been proposed by [13].

DEFINITION 5. ***BMO-Set***
The ***Best-Matches-Only*** *result set contains only the best matches w.r.t the strict partial order of a preference* P*. It is a selection of unordered result tuples where all tuples in the BMO-set are undominated by others regarding the preference* P*.*

In principle, efficient BMO query evaluation requires two new relational operators. We define

$$\sigma[P](R) := \{t \in R \mid \neg\exists t' \in R : t[A] <_P t'[A]\}$$

as **preference selection**. It finds all best matching tuples t for a preference $P = (A, <_P)$ with $A \subseteq \text{attr}(R)$[2]. If none exists, it delivers *best-matching alternatives*, but *nothing worse*.

A preference can also be evaluated in grouped mode, given some $B \subseteq \text{attr}(R)$. This can be expressed as the **grouped preference selection**

$$\sigma[P \text{ groupby } B](R) :=$$
$$\{t \in R \mid \neg\exists t' \in R : t[A] <_P t'[A] \wedge t[B] = t'[B]\}\,.$$

$\sigma[P](R)$ and $\sigma[P \text{ groupby } B](R)$ can perform the match-making process as required by BMO semantics.

[2]We use attr(R) to denote all attributes of a relation R

4. OPTIMIZATION TECHNIQUES

[11, 15] and [3] laid the foundations of a framework for preference query optimization that extends established query optimization techniques from relational databases. They presented a variety of new laws for preference relational algebra to optimize preference queries.

The first key to our algebraic optimization is a **dominance criterion**, which allows us to neglect dominated tuples which do not satisfy the user's preferences, before evaluating the cartesian product. The second optimization are additional **selection operators** which can be retrieved from the hard constraints and even applied before building the cartesian product. This speeds-up evaluation and reduces memory and computation costs.

4.1 Dominance Criterion

In [5] we introduced a dominance criterion to optimize *Preference SUM-constraint queries*, but this criterion is only valid for a single sum constraint in the query. Now we consider more complex hard constraints (e.g. calories, vitamin C and fat in example 1) and therefore we have to modify this dominance criterion given in [5]. But the dominance criterion is not restricted to *sum* constraints, it is also valid for multiplication, or, generally for each **monotone function**. Before we give the extended definition of the dominance criterion, we define the class of preference queries our approach can handle.

DEFINITION 6. ***Preference Query with Multiple Constraints***
We define a ***Preference Query with Multiple Constraints*** *as* [3]

$$\mathbf{Q} := \sigma[P_1 \,\Phi_1 ... \,\Phi_r \, P_r]\, \sigma_F\, (R_1 \times ... \times R_r)$$

where

$$F := F_1 \wedge ... \wedge F_r$$

are multiple constraints with

$$F_k = \sum_{i=1}^{r} \rho_i \cdot A_{i_j} \,\Theta_k\; c_k, \quad j = 1, ..., n_i \;\;,$$

a ***SUM-constraint*** *or*

$$F_k = \prod_{i=1}^{r} \rho_i \cdot A_{i_j} \,\Theta_k\; c_k, \quad j = 1, ..., n_i \;\;,$$

a ***Multiplication-constraint*** *with*

- $P_i = (B_i, <_{P_i})$ *arbitrary preferences*
- $\Phi_i \in \{\otimes, \,\&\}, \;\; i = 1, ..., r$ *a* ***Pareto*** *or* ***Prioritization*** *operator (cp. section 3)*
- $R_i(A_{i_1}, ..., A_{i_{n_i}}, B_i)$, $i = 1, ..., r$ *database relations, where* n_i *is the number of numerical attributes in relation* R_i
- A_{i_j} *positive numerical attributes*
- $\rho_i \in \mathbb{R}_0^+$, $i = 1, ..., r$ *a numerical multiplicator*
- $\Theta_k \in \{<, \leq, >, \geq, =, \neq\}$,
- $c_k \in \mathbb{R}_0^+$ *a* ***constant***

[3]$\sigma[P]$ means preference selection, i.e. a soft constraint, whereas σ_F denotes a classical relational algebra selection, i.e. a hard constraint.

Note: The definition can also be extended to queries with join conditions, see section 4.3.

Now we provide the first optimization technique for preference queries with multiple constraints.

THEOREM 1. ***Extended Dominance-Criterion***
Consider a query

$$\mathbf{Q} := \sigma[P_1 \ \Phi_1 ... \ \Phi_r \ P_r] \ \sigma_F \ (R_1 \times ... \times R_r)$$

as described in definition 6.
Let tuples $t, t' \in R_i$ such that

$$t[B_i] <_{P_i} t'[B_i] \quad \wedge \qquad t[A_{i_1}] \ \hat{\Theta}_1 \ t'[A_{i_1}] \ \wedge \ ... \ \wedge \ t[A_{i_r}] \ \hat{\Theta}_r \ t'[A_{i_r}] \qquad (*)$$

and $\hat{\Theta}_j, \ \ j = 1, ..., r$ defined as

$$\hat{\Theta}_j := \begin{cases} \geq & \textit{iff} \quad \Theta_j \in \{\leq, <\} \\ \leq & \textit{iff} \quad \Theta_j \in \{\geq, >\} \\ = & \textit{iff} \quad \Theta_j \in \{=, \neq\} \end{cases}$$

*Then an optimal solution exists for our Preference Query with Multiple Constraints Q **without** the tuple $t \in R_i$, i.e. using the dominance criterion $(*)$ for each relation, Q leads to a correct and complete solution.*

PROOF. Let $w := (t_1, ..., t, ..., t_r)$ and $v := (t_1, ..., t', ..., t_r)$ two tuples in $R := R_1 \times ... \times R_r$ which only differ in t and t' with $t, t' \in R_i$ and

$$t[B_i] <_{P_i} t'[B_i] \quad \wedge \qquad t[A_{i_1}] \ \hat{\Theta}_1 \ t'[A_{i_1}] \ \wedge \ ... \ \wedge \ t[A_{i_r}] \ \hat{\Theta}_r \ t'[A_{i_r}]$$

Then, it is evident that:

1) if one of the hard constraints F_k in F fails for v, also the hard constraint fails for w, since $t[A_i] \ \hat{\Theta}_j \ t'[A_i]$ (**we only consider monotone functions**). Therefore w is not an element of the solution.
2) if v fulfills all hard constraints, then
   a) if w fails one hard constraint in F, then w is not an element of the solution.
   b) if w also fulfills all hard constraints in F, then we know $t[B_i] <_{P_i} t'[B_i]$, i.e. tuple t' is preferred to t.
   Now, it follows from the preference $P := P_1 \ \Phi_1 ... \ \Phi_r \ P_r$ that v is preferred over w since t' is preferred w.r.t P_i and all others are equal.

□

Remarks: With the SUM-constraints respectively the Multiplication-constraints F_k in definition 6 we specified a wide range of monotone arithmetic functions possible in a *SELECTION* clause of a Preference SQL query. However, the dominance criterion is valid for **all monotone functions**, since the proof only refers to monotone functions and the given preferences.

We will give a short example for the extended dominance criterion.

EXAMPLE 3. *Revisit example 1 with Mrs. Diet's hard constraints. For her query we have three SUM-constraints with maximal 1100 kcal (Θ_1 is $\leq$), at least 38g of Vitamin C (Θ_2 is $\geq$) and maximal 9g fat (Θ_3 is $\leq$). Further on, she prefers chicken soup. Table 1 represents a simple soup relation.*

Table 1: Example for the dominance criterion

Soups	*ID*	*Name*	*Cal*	*Vc*	*Fat*
	S1	Vegetable	59	12	1
	S2	Chicken	110	2	4
	S3	Chicken	198	9	2
	~~S4~~	~~Noodle~~	~~453~~	~~1~~	~~17~~

Since 'S4' has more calories than 'S3' ($\hat{\Theta}_1$ is $\geq$), less Vitamin C than 'S3' ($\hat{\Theta}_2$ is $\leq$), more fat than 'S3' ($\hat{\Theta}_3$ is $\geq$) and 'S4' is worse than 'S3' concerning the soup preference, tuple 'S4' is dominated, i.e. we have not to consider 'S4' in the cartesian product and the hard selection. In the preference evaluation tuple 'S4' is unimportant, since 'S2' and 'S3' are preferred, even if the combination with 'S4' would fulill all hard constraints. This results in the undominated tuples of {'S1','S2','S3'}. Tuple 'S3' is not worse than 'S2' with regard to the preference. Tuple 'S1' is worse than 'S2' and 'S3' concerning the same preference, but maybe 'S2' and 'S3' do not fulfill the hard constraints ('S1' has less calories, higher vitamin C and less fat). So we must take into account tuple 'S1'.

As shown in the example, the dominance criterion from theorem 1 allows us to eliminate tuples from the relations before building the cartesian product. This reduces relation sizes and therfore speeds-up the computation.

For evaluation of the dominance criterion we use the BMO query model and apply the so called CUTOFF preference constructor, also see [5]. Since we extended the definition of the dominance criterion, we also have to change the CUTOFF constructor for a valid BMO evaluation.

DEFINITION 7. ***CUTOFF Preference Constructor***
Given a preference $P = (B, <_P)$ on a relation $R(A_1, ..., A_n, B)$ with tuples $x = (x_1, ..., x_n, x_B), \ y = (y_1, ..., y_n, y_B)$ in a Preference Query with Multiple Constraints (cp. definition 6).
Then $P_c := CUTOFF(P)$, if:

$$x <_{P_c} y \quad \textit{iff} \quad x_B <_P y_B \quad \wedge \quad x_1 \ \hat{\Theta}_1 \ y_1 \ \wedge \ ... \ x_n \ \hat{\Theta}_n \ y_n$$

where

$$\hat{\Theta}_j := \begin{cases} \geq & \textit{iff} \quad \Theta_j \in \{\leq, <\} \ \ j = 1, ..., n \\ \leq & \textit{iff} \quad \Theta_j \in \{\geq, >\} \ \ j = 1, ..., n \\ = & \textit{iff} \quad \Theta_j \in \{=, \neq\} \ \ j = 1, ..., n \end{cases}$$

*Tuple x is worse than y concerning the CUTOFF-preference P_c, if y_B is better than x_B regarding the given preference P (the preference specified by the user) **and** all comparison operators $\hat{\Theta}_j$ are evaluated to true.*

The evaluation of the dominance criterion by the CUTOFF constructor makes the computation of Preference Queries with Multiple Constraints evident: Our Preference SQL engine [16] applies the CUTOFF preference independently on each stream of tuples. Afterwards it performs the cartesian product (checking the hard sum constraint) and finally evaluates the preference selection by a skyline algorithm [2, 21, 20], also cp. section 4.3.

4.2 Selection Operators

In this section we add hard selection constraints to the original Preference SQL query to filter tuples before building the cartesian product. The basic idea was developed by [19] and is now adapted to preference queries with multiple constraints. For this, each constraint in the query condition is rewritten as a set of simpler constraints and an additional hard **selection operator** is applied on each relation.

4.2.1 Selection Operator for SUM-Constraints

Consider a sum constraint like

$$\rho_1 A_1 + ... + \rho_r A_r \; \Theta \; c \qquad (**)$$

with c a constant, $\Theta \in \{<, >, \leq, \geq\}$ and $\rho_k \in \mathbb{R}^+$. Obviously

$$\rho_i A_i \;\; \Theta \;\; c - \sum_{k=1, k\neq i}^{r} \rho_k A_k \quad \text{for } i = 1, ..., r$$

From this expression we can calculate the lower (lb_i) and upper (ub_i) bounds of it as follows:

$$lb_i = c - \sum_{k=1, k\neq i}^{r} max(\rho_k A_k) \quad \text{for } i = 1, ..., r$$

$$ub_i = c - \sum_{k=1, k\neq i}^{r} min(\rho_k A_k) \quad \text{for } i = 1, ..., r$$

Using lb_i and ub_i it is possible to create additional constraints for attribute A_i depending on $\Theta \in \{<, >, \leq, \geq\}$. If $\Theta \in \{<, \leq\}$ we call such constraints **maximum sum constraints** and derive

$$A_i \; \Theta_{<,\leq} \; \frac{ub_i}{\rho_i} \; .$$

For $\Theta \in \{>, \geq\}$ we call such constraints **minimum sum constraints** and derive the following additional constraint for A_i:

$$A_i \; \Theta_{>,\geq} \; \frac{lb_i}{\rho_i} \; .$$

Any value which does not fulfill the additional constraint can not satisfy the sum constraint shown in ($**$). Through these additional hard constraints we can filter out tuples from the relations that will not lead to any query result due to the hard constraints before building the cartesian product.

EXAMPLE 4. *Consider the example database in table 2 and Mrs. Diet's preference query*

```
SELECT S.name, M.name, B.name
FROM   Soups S, Meats M, Beverages B
WHERE  S.cal + M.cal + B.cal ≤ 1100
  AND  S.Vc + M.Vc + B.Vc ≥ 38
  AND  S.fat + M.fat + B.fat ≤ 9
PREFERRING
  S.name IN ('Chicken soup')        AND
 (M.name IN ('Beef')
          AND M.Cholesterol LOWEST) AND
  B.name IN ('Red wine')
```

For the hard constraint of 1100 kcal we can express the following upper bounds and therefore maximum constraints:

- $min(M.Cal) + min(B.Cal) = 903 \Rightarrow S.Cal \leq 197$
- $min(S.Cal) + min(B.Cal) = 144 \Rightarrow M.Cal \leq 956$
- $min(S.Cal) + min(M.Cal) = 877 \Rightarrow B.Cal \leq 223$

Table 2: Example database

	Soups	*ID*	*Name*	*Cal*	*Vc*	*Fat*
		S1	Vegetable	59	12	1
$S.Vc < 3$		~~S2~~	~~Chicken~~	~~140~~	~~2~~	~~4~~
$S.Cal > 197$		~~S3~~	~~Chicken~~	~~198~~	~~9~~	~~2~~
$DC, S.Cal > 197$		~~S4~~	~~Noodle~~	~~453~~	~~1~~	~~17~~

	Meats	*ID*	*Name*	*Cal*	*Vc*	*Fat*	*Cholesterol*
		M1	Turkey	818	13	8	6
		M2	Beef	857	14	6	4
DC		~~M3~~	~~Pork~~	~~911~~	~~12~~	~~12~~	~~15~~

	Beverages	*ID*	*Name*	*Cal*	*Vc*	*Fat*
$B.Vc < 12$		~~B1~~	~~Red Wine~~	~~85~~	~~1~~	~~0~~
		B2	Red Wine	181	14	0
		B3	Coke	220	21	2
$B.Cal > 223$		~~B4~~	~~Lemonade~~	~~281~~	~~17~~	~~2~~
$B.Cal > 223$		~~B5~~	~~Red Wine~~	~~190~~	~~4~~	~~0~~

This means, there only exists a solution for the hard sum constraint with tuples in the soup relation with $S.Cal \leq 197$. Tuples with $S.Cal > 197$ ($\{$S3, S4$\}$) can be eliminated from the relation (take not part in the cartesian product) since there is no combination with less than 1100 kcal possible. The same with beverages. In the meats relation all tuples have $M.Cal \leq 956$ and therefore none can be eliminated by additional selection operators. But tuple 'M3' is dominated due to the dominance criterion (theorem 1), see table 2. For the hard constraint of 38 g of Vitamin C it follows: $S.Vc \geq 3$, $M.Vc \geq 5$ and $B.Vc \geq 12$. We can dominate tuple 'S2' ($S.Vc = 2$ and it should be $S.Vc \geq 3$), and tuple 'B1'. For the fat constraint there are no valid additional selection operators.

Considering this simple database we reduced the cartesian product from 60 possible combinations to 4 combinations. Afterwards it is possible to check all hard constraints and thereafter apply a skyline algorithm to evalute the user preference on the remaining combinations which leads to the result $\{$S1, M2, B2$\}$.

4.2.2 Selection Operator for Multiplication Constraints

We can also introduce selection operators for multiplication constraints. This is analogous to the SUM constraints. Consider

$$\rho_1 A_1 \cdot ... \cdot \rho_r A_r \; \Theta \; c \qquad (***)$$

with c a constant, $\Theta \in \{<, >, \leq, \geq\}$ and $\rho_k \in \mathbb{R}^+$. We again calculate the lower (lb_i) and upper (ub_i) bounds of it as follows:

$$lb_i = c \cdot \left(\prod_{k=1, k\neq i}^{r} max(\rho_k A_k) \right)^{-1} \quad \text{for } i = 1, ..., r$$

$$ub_i = c \cdot \left(\prod_{k=1, k\neq i}^{r} min(\rho_k A_k) \right)^{-1} \quad \text{for } i = 1, ..., r$$

We get the **maximum multiplication constraint** for $\Theta \in \{<, \leq\}$

$$A_i \; \Theta_{<,\leq} \; \frac{ub_i}{\rho_i}$$

and the **minimum multiplication constraint** for $\Theta \in \{>, \geq\}$

$$A_i \; \Theta_{>,\geq} \; \frac{lb_i}{\rho_i} \; .$$

Depending on the constraints we can eliminate tuples from the relations to speed-up evaluation.

4.3 Rewriting Technique

Hafenrichter and Kießling [11, 15] constructed a preference query optimizer as an extension of an existing SQL optimizer, adding new heuristics like 'push preference'. This query optimizer is based on a classical Hill-Climbing algorithm given by Ullmann [23]. With the preparatory work from section 4 we can develop transformation laws for preference relational algebra that allow us to eliminate dominated tuples before building the cartesian product by *inserting the CUTOFF preference* and the *additional selection operators* into the query.

We intergrated our novel rewriting techniques - the *dominance criterion* (section 4.1) and the additional *selection operators* (4.2) - into this preference query optimizer.

For this, we refer to the notation introduced in section 4, this means, we skip some formal definitions and keep it short and clearly arranged. We abbreviate the use of the CUTOFF(P) constructor for a preference P with $\mathbf{P_c}$ and the selection operators with $\mathbf{SO}$. Since figures are more meaningful than sentences, we demonstrate the rewriting technique using an operator tree.

With the help of P_c and the selection operators SO we can transform a Preference Query with Multiple Constraints into an optimized query as follows:

i) Determine the lower respectively upper bounds (abbr. b_i) for each constraint. Insert the corresponding selection operators $\sigma_{A_i\ \Theta_i\ b_i}$ into the query tree.

ii) Apply the CUTOFF preference constructor P_c to the result of the selection operator.

iii) Build cartesian product and check the constraints $F_1 \wedge ... \wedge F_r$.

iv) Evaluate the preference selection $P_1\ \Phi_1 ...\ \Phi_r\ P_r$ by a skyline algorithm [4].

Formal:

COROLLARY 1. ***Insert $\mathbf{P_c}$ and SO into Cartesian Product*** *Consider a preference query*

$$Q := \sigma[P_1\ \Phi_1 ...\ \Phi_r\ P_r]\ \sigma_F\ (R_1 \times ... \times R_r)$$

where $F := F_1\ \wedge\ ...\ \wedge\ F_r$ *and*

- $P_i = (B_i, <_{P_i})$ *arbitrary preferences*
- $\Phi_i \in \{\otimes,\ \&\},\ \ i = 1, ..., r$ *a* ***Pareto*** *or* ***Prioritization*** *operator (cp. section 3)*
- $R_i(A_{i_1}, ..., A_{i_{n_i}}, B_i),\ i = 1, ..., r$ *database relations, where* n_i *is the number of numerical attributes in relation* R_i
- A_{i_j} *positive numerical attributes*
- $F_k = \sum_{i=1}^{r} \rho_i \cdot A_{i_j}\ \Theta_k\ \ c_k$ ***SUM-constraints*** *or*
- $F_k = \prod_{i=1}^{r} \rho_i \cdot A_{i_j}\ \Theta_k\ \ c_k$ ***Multiplication-constraints***
- $\rho_i \in \mathbb{R}_0^+,\ i = 1, ..., r$ *a numerical multiplicator*
- $\Theta_k \in \{<, \leq, >, \geq, =, \neq\}$, *and* $c_k \in \mathbb{R}_0^+$ *a* ***constant***
- b_i *the* ***lower*** *respectively* ***upper bounds*** *(cp. section 4.2) for the selection operators (only valid, if* $\Theta_k \in \{<, \leq, >, \geq\}$*)*

Than we can transform the query Q into the following optimized operator tree, see figure 1:

[4]Since a skyline algorithm [2, 21, 20] only can handle Pareto preferences, for Prioritization an algorithm developed by [11] can be applied.

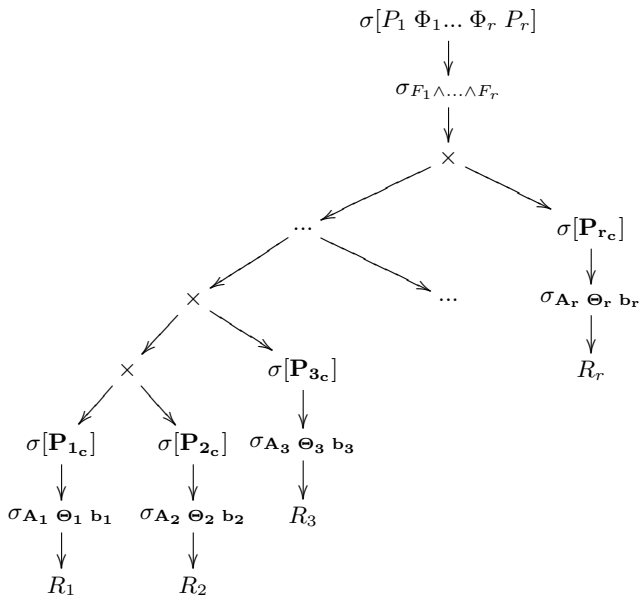

Figure 1: Optimized Preference Query

PROOF. The proof is given by theorem 1. □

Remarks:

- It is allowed to change CUTOFF and the additional hard selection operator in the Hill-Climbing algorithm. However, applying hard selection first reduces computation costs for evaluating CUTOFF.
- In the case of joins like $R_1 \bowtie_{R_1.X=R_2.X} R_2$ we have to ensure that we do not eliminate join partners, i.e. for each tuple in the first relation there must exist a join partner in the second relation. To get rid of this problem we have to evaluate the CUTOFF preference as a **grouped preference selection**, see section 3.2, [5] and [11]. Therefore, the CUTOFF preference is only evaluated for tuples in the same equivalence class, i.e. grouped by X.

5. EXPERIMENTAL RESULTS

In order to evaluate our rewriting techniques, we performed several experiments. For this we intergrated our rewriting techniques into the preference query optimizer developed by Hafenrichter [11], also cp. section 4.3. We used a real-world food database published by the USDA [24]. This database contains nutritional facts for more than 7000 types of food. From this database we created three relations as in example 1: *Soups*, *Meats* and *Beverages* containing information about their eponymous types of food. The sizes of these relations are as follows: There are 500 soups, 700 meats and 250 beverages available, i.e. about 87.500.000 possible combinations.

We run all test queries on an Oracle 10.0 database system in combination with the preference query optimizer described in [11] (Java implementation). The system is running on a Linux machine (Intel Dual Core CPU 1.6 GHz, 2GB main memory).

We evaluated the efficiency of our rewriting techniques introduced in section 4 by comparing the response times of several *Preference Queries with Multiple Constraints*. **Due to limited space** we only report a few results, with representative performance, shown below. In our result figures, we abbreviate the standard approach (full cartesian product) with **no rewriting**. Using the dominance

criterion from section 4.1 we write **DC**, using the selection operators from section 4.2 we write **SO** and for the evaluation of the dominance criterion in combination with the selection operators we write **DC+SO**.

The test query is based on example 1 and contains three constraints. For representation we only varied the amount of calories *cal*, which must be less or equal than a value called **max_cal**. The amount of vitamin C (*Vc*) and the fat value (*fat*) we fixed to the values given in example 1. This leads to the query

```
SELECT S.name, M.name, B.name
FROM   Soups S, Meats M, Beverages B
WHERE  S.cal + M.cal + B.cal ≤ max_cal
  AND  S.Vc + M.Vc + B.Vc ≥ 38
  AND  S.fat + M.fat + B.fat ≤ 9
PREFERRING
  S.name IN ('Chicken soup')        AND
 (M.name IN ('Beef')
          AND M.Cholesterol LOWEST) AND
  B.name IN ('Red wine')
```

Notice, varying the parameter **max_cal** varies the selectivity of the query, while varying the size of the relations changes the size of the problem to be solved. Therefore, we varied the required sum constraints in order to change the selectivity and we varied the size of the relations.

In our first test series we only varied the **max_cal** value in a range from 600 to 1700 calories, see figure 2.

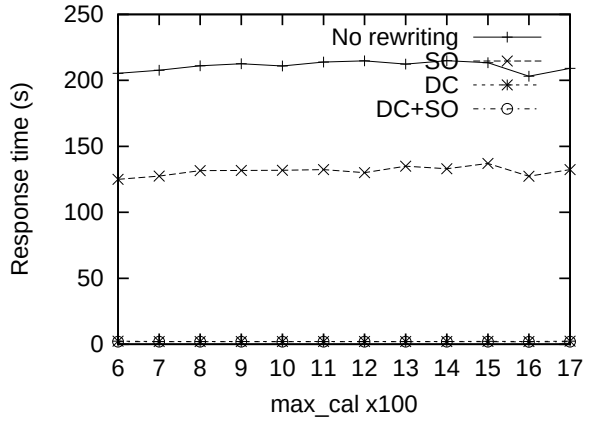

Figure 2: Performance results for different max_cal.

Since the dominance criterion only depends on the preferences and not on the hard constraints, the response time for the preference query with different **max_cal** is nearly constant for each approach. Using the selection operators (*SO*), the query response time is better than the standard approach (*no rewriting*), since some tuples can be eliminated before building the cartesian product. Using the dominance criterion (*DC*) or the combination (*DC+SO*), the response time of the query is about 2 seconds, as can be seen in figure 3, which is a zoomed figure of figure 2.

In all our tests with varied *cal*, *Vc* and *fat* it performed out, that *DC+SO* performs best for all, but is only a little bit better than *DC* alone. The reason is the worse evaluation of the additional selection operators for our real-world data from the USDA, since there are many tuples with a *NULL* or 0 (zero) entry in the corresponding attribute. For data with no or less *NULLs* and 0 entries the additional selection operators should perform much better.

Next, we run the query above with different relation sizes (but

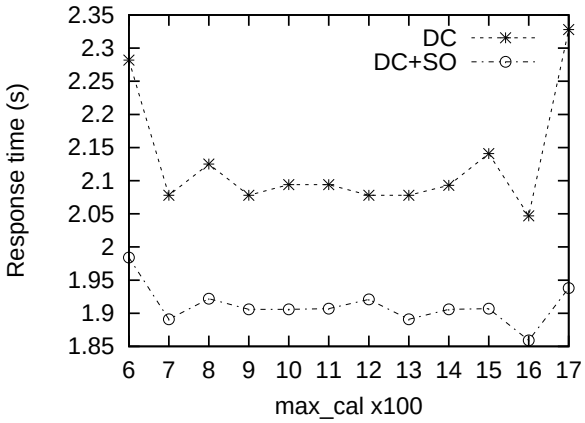

Figure 3: Comparision of DC and DC+SO.

fixed max_cal = 1100) and demonstrate the performance results in figure 4.

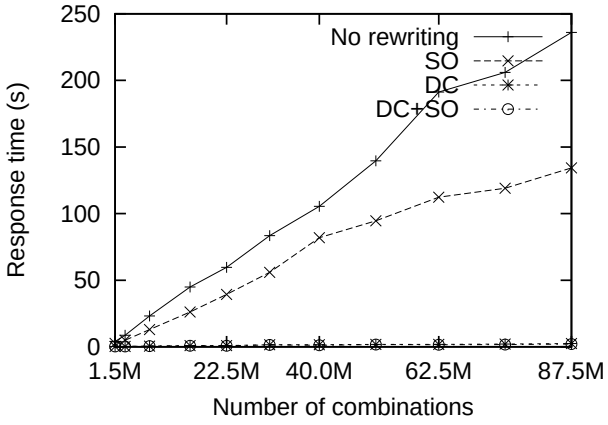

Figure 4: Performance results for different relations sizes.

As above, *DC* and *DC+SO* performs extremly good and for a comparison of these two, we show again a zoomed figure from the one above, see figure 5. The difference between *DC* and *DC+SO* is marginal, because of the reason given above (*NULL* entries).

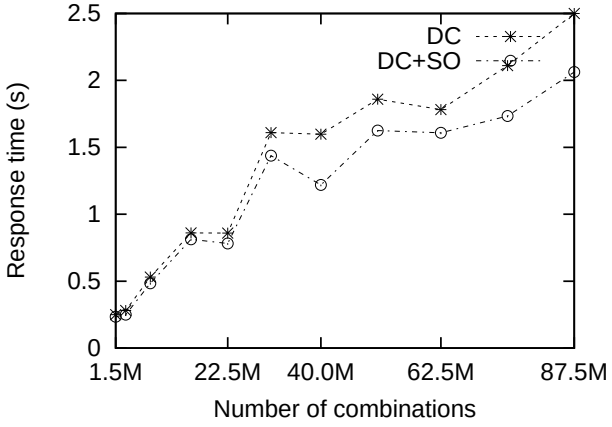

Figure 5: Comparision of DC and DC+SO.

Obviously our rewriting techniques speeds-up the evaluation because of writing in the *CUTOFF* preference and the additional selection operators into the query. The combination *DC+SO* is better than all other approaches, but the difference between *DC* and *DC+SO* is not very large.

From the experimental results of our benchmark queries, we can see that our proposed rewriting techniques improve the query performance consistently for different types of sum constraint queries.

6. SUMMARY AND OUTLOOK

Finding efficient query optimization techniques for *Preference Queries with Multiple Constraints* is beneficial for a variety of practical database applications, e.g. in planning tasks or tourism. In this paper we extended our dominance criterion to work with multiple hard constraints like sum or multiplication constraints. The dominance criterion as well as the insertion of additional selection operators based on mathematical observations eliminate tuples from relations before building the cartesian product and therefore reduces memory and computation costs, i.e. speeds-up the evaluation of *Preference Queries with Multiple Constraints*. The performance speed-ups observed so far give already strong evidence that a tightly coupled implementation inside an existing SQL query engine can achieve excellent performance.

For future work we want to develop a cost-based optimization for Pareto preference and Prioritization queries with multiple constraints. We want to develop algorithms which completly avoid the cartesian product. For this we developed a first approach which seems to be good. However, this is much more complex and needs more attention and deeper research.

7. REFERENCES

[1] P. K. Agarwal, L. Arge, J. Erickson, P. G. Franciosa, and J. S. Vitter. Efficient Searching with Linear Constraints. In *PODS*, pages 169–178, 1998.

[2] S. Börzsönyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *Proceedings of the 17th International Conference on Data Engineering (ICDE)*, pages 421–430. IEEE Computer Society, 2001.

[3] J. Chomicki. Preference Formulas in Relational Queries. In *ACM Transactions on Database Systems (TODS)*, volume 28, pages 427–466. ACM Press, 2003.

[4] S. Döring, T. Preisinger, and M. Endres. Advanced Preference Query Processing for E-Commerce. In *Proceedings of 23rd Annual ACM Symposium on Applied Computing (SAC)*, pages 1457–1462. ACM, 2008.

[5] M. Endres and W. Kießling. Optimization of Preference Queries under Hard Sum Constraints. In *Proceedings of the 4th Multidisciplinary Workshop on Advances in Preference Handling (AAAI). Chicago, Illinois (USA)*, 2008.

[6] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS)*, pages 102–113. ACM, 2001.

[7] B. Faltings, P. Pu, and J. Zhang. Agile Preference Models based on Soft Constraints. In *Challenges to Decision Support in a Changing World*, 2005.

[8] P. Georgiadis, I. Kapantaidakis, V. Christophides, E. M. Nguer, and N. Spyratos. Efficient rewriting algorithms for preference queries. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, México*.

[9] J. Goldstein, R. Ramakrishnan, U. Shaft, and J.-B. Yu. Processing Queries By Linear Constraints. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona*, pages 257–267. ACM Press, 1997.

[10] S. Guha, D. Gunopoulos, N. Koudas, D. Srivastava, and M. Vlachos. Efficient approximation of optimization queries under parametric aggregation constraints. In *Proceedings of the 29th international conference on Very large data bases (VLDB)*, pages 778–789. VLDB Endowment, 2003.

[11] B. Hafenrichter and W. Kießling. Optimization of Relational Preference Queries. In *The 16th Australasian Database Conference (ADC)*, pages 175–184, 2005.

[12] I. F. Ilyas, W. G. Aref, and A. K. Elmagarmid. Supporting Top-k Join Queries in Relational Databases. In *Proceedings of the 29th international conference on Very large data bases (VLDB)*, pages 754–765. VLDB Endowment, 2003.

[13] W. Kießling. Foundations of Preferences in Database Systems. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 311–322, 2002.

[14] W. Kießling. Preference Queries with SV-Semantics. In *Proceedings of the 11th International Conference on Management of Data (COMAD)*, pages 15–25, 2005.

[15] W. Kießling and B. Hafenrichter. Optimizing Preference Queries for Personalized Web Services. In M. H. Hamza, editor, *Communications, Internet, and Information Technology*, pages 461–466. IASTED/ACTA Press, 2002.

[16] W. Kießling and G. Köstler. Preference SQL - Design, Implementation, Experiences. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 990–1001, 2002.

[17] C. Liu, L. Yang, and I. Foster. Efficient Relational Joins with Arithmetic Constraints on Multiple Attributes. In *Proceedings of the 9th International Database Engineering & Application Symposium (IDEAS)*, pages 210–220. IEEE Computer Society, 2005.

[18] C. Liu, L. Yang, I. Foster, and D. Angulo. Design and evaluation of a resource selection framework for grid applications, 2002.

[19] S. Nestorov, C. Liu, and I. T. Foster. Efficient Processing of Relational Queries with Sum Constraints. In *APWeb/WAIM*, pages 440–451, 2007.

[20] T. Preisinger and W. Kießling. The Hexagon Algorithm for Evaluating Pareto Preference Queries. In *Proceedings of the Multidisciplinary Workshop on Advances in Preference Handling (VLDB)*, 2007.

[21] T. Preisinger, W. Kießling, and M. Endres. The BNL++ Algorithm for Evaluating Pareto Preference Queries. In *Proceedings of the Multidisciplinary Workshop on Advances in Preference Handling (ECAI)*, 2006.

[22] M. Stonebraker and J. M. Hellerstein. Content integration for e-business. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data (SIGMOD)*, pages 552–560. ACM, 2001.

[23] J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume I*. Computer Science Press, 1988.

[24] USDA. USDA national nutrient database for standard reference. http://www.nal.usda.gov/fnic/, 2007.

Context-Based Personalization for Mobile Web Search

Mario Arias
GRINBD. Univ. of Valladolid
47007 Valladolid, Spain
mario.arias@gmail.com

José M. Cantera
Telefónica I+D
47151 Boecillo, Spain
jmcf@tid.es

Jesús Vegas
GRINBD. Univ. of Valladolid
47007 Valladolid, Spain
jvegas@infor.uva.es

ABSTRACT

User experience while searching for web pages on the move can be far from satisfactory due to the inherent limitations of the input modes available in mobile devices. On the other hand, end-users can benefit from the availability of context-aware information anywhere, anytime. To overcome the usability problem and exploit context information at the same time, we propose a thesaurus-based *semantic context-aware autocompletion* mechanism. Our system can help the user in completing the desired query terms avoiding manual typing. In addition we are capable of filtering out non-relevant query terms for the Context in which the search process is conducted. Our context-aware proposal is based on a model which represents formally all the information about the user circumstances, the access mechanism (device and web browser) and the surrounding environment. Our evaluation reveals that users can find new relevant context-aware results with less effort.

1. INTRODUCTION

Mobile devices have evolved to provide bigger full-color screens, enhanced processing power and faster and permanent broadband Internet connections. These technologies have brought the World Wide Web to mobile devices introducing new requirements and expectations. Nonetheless, the vast majority of web sites and search engines are usually designed with desktop computers in mind. For that reason, current mobile search experience is far from satisfactory [17]. Search engine analysts, being aware of this problem, have designed mobile-oriented views to provide the same service from a smaller interface. Content transformation (reformatting) proxies have been devised to reduce pages on the fly [8], making them more accessible for the mobile web. Such approach deals with one of the major limitations of the mobile web, screen size, but it does not address the problem of the limited input modes. Furthermore, it does not take advantage of the contextual information available in a mobile environment in order to provide more accurate results.

VLDB '08, August 24-30, 2008, Auckland, New Zealand

Deficient handset's input modes are one of the most important limitations when using a mobile search engine. According to recent studies [9], users need an average of 40 key-presses with a 12-key keypad, and about 40 seconds to enter a query with a cell phone. That is a too big effort, users get easily tired and sometimes even give up. This leads to the *mismatched query problem* [5], mobile queries become even shorter and more ambiguous than the same queries in a desktop environment. The immediate consequence is that they do not reflect user's intentions precisely, hence the obtained results are poor. Ultimately the user is forced to visit a big number of useless web sites and to navigate through several result sets in order to find the needed information. This search flow might be acceptable when surfing the web from a desktop computer, but it becomes an extremely arduous task when performed from a mobile device.

Mobile Web Search introduces new challenges not present in traditional web search. Users normally own modern cell phones which allows them to be permanently online anywhere, anytime. A typical mobile web search scenario consists of a user outdoors with an information need. At this point he takes his phone and uses a web search engine to find an answer to a query. Furthermore, he is probably doing something else at the same time, like walking or talking to a friend. In such situation the user needs a short, fast but also accurate answer to his query.

Autocompletion mechanisms appear as a natural improvement to the limited input modes problem. In fact, any modern information retrieval system can obtain the most popular query terms or phrases from indexed pages. Such elements can be used to interactively complete user's queries. An autocompletion engine can be helpful both by saving typing time and by finding new, serendipitous terms. However a pure syntactic approach can present difficulties, because it will be based on the coincidence of string representations (words), but not in concepts and their relationships. The corollary is that, the autocompletion mechanism will be helpful if the user intends to use the same word that the system is expecting, but if, for instance, a synonym is in user's mind, the system will be unable to recognize and autocomplete it. The previous facts have led us to the introduction of a concept-driven *semantic and context-aware autocompletion engine.*

Initially, we have created a thesaurus which represents concepts together with their synonyms and relationships. We have modeled concepts corresponding to the domain of services offered to the public, such as hotels, restaurants, tourism offices, public transport stations and similar. We

have chosen such set of concepts, as they are good candidates to be the most likely-to-be-used query terms in a mobile environment (where the user is always looking for something to address a very specific and punctual necessity). For example, our thesaurus includes the concept "dinner" related to "food", "restaurant" and "supermarket".

An autocompletion mechanism targeted to the mobile environment can also benefit from the availability of contextual information. For example, there is no point in suggesting the term "beach" while in a place which it is not on the seaside, or during winter time. To deal with those scenarios, we have extended our semantic autocompletion engine with context-aware recommendation, which filters out non-contextually-relevant concepts. As a result we are able to suggest the best query terms according to each situation. To implement such advanced features we have introduced a formal Context Model which represents all the significative properties about the environment (place, access mechanism, user profile, etc.) in which the search process is conducted.

We have integrated our semantic and context-aware autocompletion system into a working prototype. Such prototype also includes a novel user interface designed to meet all the requirements imposed by the mobile environment. Finally, we have tested the feasibility of the system by making a qualitative analysis of the improvement obtained in the user experience.

The rest of this paper is organized as follows. Section 2 briefly describes the background and related work regarding context awareness, personalization and recommender systems. The detailed description of our proposal can be found on section 3. The evaluation and experimentation results are detailed on section 4. Finally section 5, is devoted to summarize the conclusions and the guidelines for future work.

2. BACKGROUND AND RELATED WORK

The expansion of embedded and handheld devices has promoted the research on the benefit of contextual information to improve Human-Computer Interaction, Ubiquitous Computing and Web Search experience [3].

A.K.Dey [2] suggests the following general definition of Context and context-awareness: *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves. A system is context-aware if it uses the Context to provide relevant information and/or services to the user, where relevancy depends on the users task.* According to this definition, the Context could be considered the current data displayed on the screen, the surrounding environment or even the whole application. For that reason every context-aware application must explicitly specify what information is part of the Context. Additionally, the Context is inherently dynamic and constantly changing. A few properties will be barely modified over time, like user name or age, but other ones may frequently vary, like time or position. We must ensure that when context properties are queried they are all up to date, or at least marked as expired [18].

The main challenge introduced by context-awareness is to come up with a flexible and unambiguous representation of the Context, including a framework to ease the development of context-aware applications. Based on this Context Model, applications can figure out which actions should be triggered or what data is potentially relevant to the user. Nonetheless, the heterogeneous nature of context-aware applications makes it impossible to have a universal, unique representation of the Context. However, a good compromise can be achieved if context models are able to manage a set of universal properties, useful for any application, in conjunction with application-specific, custom properties. Previous works [18] propose a Context taxonomy and a framework which define several abstract layers of knowledge. Such layers are ready to be mapped with the properties that actually will model the Context. In addition, formal reasoning techniques can be employed to create derived properties based on those which are directly fetched.

Another important issue has to do with gathering all the significative context information that can be of interest to an application. It is noteworthy that each application will be only interested in a limited subset of the Context. As a result, context frameworks should provide mechanisms to allow applications to express their interest in certain context properties. Additionally the context framework should be prepared to create bindings between context properties and the corresponding information sources. Such bindings should hide applications from the lower-level protocols or services used to actually obtain the context property values. Besides, some directly measured context properties are not suitable to use *as-is*; there can be properties that need a previous transformation or processing to be useful for an application. For example, an application might require GPS coordinates as input in some cases, but in others it might need the place name. As a consequence, a context framework should perform automatic context data transformations on behalf of the application.

To know which are the interests of each user it is needed to generate a user profile. The first option is to explicitly ask what are user's interests with a small survey form. This method is known to be *high quality* if the survey is correctly designed, but in general users dislike filling forms, especially when the benefits are not immediately obvious. In addition, users do not feel safe by submitting personal information to untrusted servers [11].

Collaborative filtering techniques analyze user behavior to create implicit user profiles. They take into account which links are more frequently clicked or the time spent on each one to extrapolate user interests without them even noticing. Users with similar interests can also be grouped into classes. Once the individual has been classified, assumptions about his interests can be made based on those of the whole group. This introduces the concept of social profiling, which can be valuable when a new user connects to the application for the first time and fewer details about him are available.

Commercial search engines, such as Google or Yahoo, incorporate context-independent autocompletion mechanisms intended to work in the desktop environment. Web based recommender systems have been successfully applied to modern online shopping sites, filtering and suggesting between a huge amount of available products [16]. Wietsma [20] developed a PDA-based prototype of a recommender system. Other works [10] have applied recommender systems to simplify mobile web search by offering syntactic query suggestions. However, none of them has combined semantic autocompletion mechanisms with context-aware recommendation in a mobile environment, which it is the main contribution of our research.

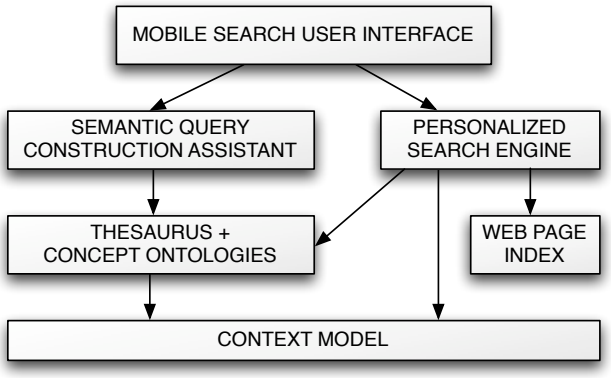

Figure 1: Mobile Web Search Framework.

Improving web search by exploiting contextual information has been previously studied, although there is no agreement on context interpretation and scope. There are authors that consider the history of visited pages as web search context. Following that criteria, they try to analyze and extract the most important keywords found in those pages, and use them to ensure that further queries will keep on subject [12, 4]. While these solutions can improve web search on the desktop, mobile search should go beyond, considering additional variables like location, conditions of the environment, points of interest nearby, weather situation or guessed user intentions [5].

Query expansion and substitution is a technique that can be used to include context information during the search process. The mechanism works as follows: Just before launching the query it is completed with additional terms [5, 13]. Such terms can be synonyms, disambiguating subject-related words or special keywords aimed at clarifying the intention, like *how to*, *ways to*, *what is*. Another way to contextualize search is by re-ranking the obtained result set according to certain properties: subject, proximity to user's location or intentions.

The use of the above methods will ensure that the most contextually-relevant results will appear on top of the result set, thus reducing reducing the number of interactions needed to find proper results [19, 11].

3. OUR PROPOSAL

To overcome previously described limitations, we have designed a framework to combine different paradigms and novel ideas from several sources, separated in different modules which work together. In first place we need to gather as much information as possible from user and environment. Then, based on that information, a recommender system selects the most relevant suggestions for user guidance. We have identified the following modules (See figure 1):

- A **Context Model** which provides a formal representation of the user, the environment and the access mechanism, enabling personalization and context-awareness.
- A **Thesaurus**, which models semantic relationships (synonym, broader, narrower, related) among the specific concepts managed by the search system.
- An **Ontology** which describes additional properties that each thesaurus concept may have. They are used to offer additional options to let the user choose instead of type.
- A **Semantic Query Construction Assistant** intended to recommend best-suitable options in context to avoid users from typing, and thus personalizing the search process.
- A **Personalized Search Engine** which makes use of all information available (the query, selected thesaurus concept, ontology values) to find relevant results. To obtains results it makes use of a traditional web search engine which maintains a web page index.
- A **User Interface** prototype adapted to the peculiarities of mobile devices, which makes use of all previous modules to provide an easier user interaction.

Our system is capable of suggesting search terms based on concepts of the thesaurus instead of words. The outcome is a more abstract level of recommendation as the system adapts to the user ideas, decoupling the system from any specific vocabulary. This solution ameliorates traditional approaches, which are restricted to syntax comparisons. Furthermore, we filter out concepts that are unlikely to be chosen in that specific concept, so less options are recommended, just the most relevant ones. In addition, each thesaurus concept is complemented with an ontology describing additional properties of that idea. Once the user have selected a specific concept, additional options are automatically presented so he just have to choose to refine his query instead of typing. For instance, if the user has selected the concept "Restaurant", a form with two fields "price" and "type of cuisine" will automatically appear.

3.1 Context Model for Personalization

The foundation for a context-aware application is a formal context specification which maintains all properties and provides a standard access to them. We used an ontology to describe all entities of the domain together in a taxonomy. We have decided to use the OWL language [15] as it is a widely accepted industry standard and therefore there are several open source tools (parsers, reasoners, editors, third-party ontologies) available to ease development.

We propose an extensible Context Model divided into three layers (Figure 2):

- **Directly Fetched Properties**. These are properties that can be automatically gathered from context information sources. For example the location coordinates obtained from a GPS sensor or the born year directly specified by the user.
- **Derived Properties**. These are "implicit" properties that can be inferred or calculated from other properties. They constitute a higher abstraction layer on top of the directly fetched properties, so they are more easily comparable against application-level ideas. For example, taking into account the location coordinates provided by a GPS, the system can obtain the name of the country and region from a GIS service, or it can calculate the user's age based on his born year.

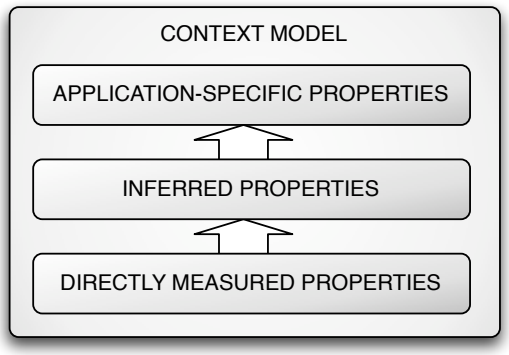

Figure 2: Context Layered Model.

- **Application-Specific Properties**. Applications might want to add or redefine properties by using their own rules, in order to extend the base context definition to better fit concrete application needs. For example, an application can define a property called *nice day* given the temperature and the atmospheric condition. All these definitions are identified at application design time, but at some point it may be required to differentiate what is a nice day according to the country. This operation can be easily accomplished by just modifying the corresponding rule.

The SWRL [6] language has been used for rule definition as it is the natural extension to OWL when rule-based depictions are needed. SWRL provides syntax and semantics to define rules affecting OWL classes, properties and individuals.

For example, we can define the following rule to create the abstract property called *niceday*:

$$weather(?w) \wedge temperature(?w, ?t) \wedge greater(?t, 20) \wedge sunny(?w, true) \rightarrow niceday(?w, true)$$

where $?w$ refers to a OWL individual which belongs to the *weather* OWL class, *temperature* is a property which links a weather instance to its temperature, *greater* is a SWRL built-in function, *sunny* is a boolean property, and *niceday* is the new defined property.

Once the context framework is defined, we proceed to identify the relevant classes, properties and relationships for any context-aware mobile web search application:

- *User Profile* This category contains all the implicit and explicit properties related to the user and his circumstances. We model several properties such as the preferred language, the date of birth, the place where he lives, etc. A FOAF [1] extension is used for user profile modeling.
- *Device and Browser* This set of properties describe the characteristics of the user's device and web browser. Such properties make it possible to distinguish between a cell phone and a PDA, or between a rich AJAX-enabled browser and a first generation limited browser. Detailed information is crucial in order to take full advantage of hardware capabilities, for example devices with bigger screens can show more rows of a table at the same time than those with smaller ones.

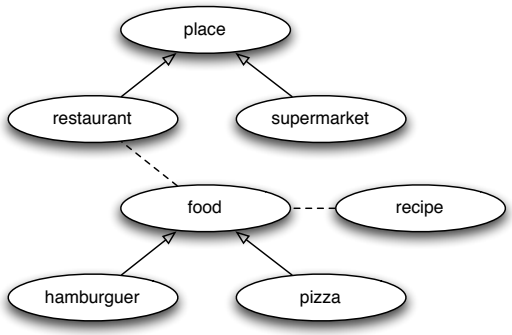

Figure 3: Thesaurus concept lattice example.

- *Geospatial Context* We model the current user location not only in terms of geographic coordinates but also in terms of the kind of place where the user is (on the seaside, on the countryside, at an airport or station, etc.). Such contextual information can be later used to promote the tasks which are more typical in those certain situations.
- *Environment Conditions* These are properties that model the surrounding environment. In our prototype we model mainly the weather conditions gathered from Web Services published on the Internet. For example, if the user is searching for leisure activities and the weather is rainy, our recommender system should suggest theater or cinema, rather than outdoor activities.
- *Date and Time* They are useful to restrict behavior at certain times or dates. For instance, there is no point in going to the cinema at eight o'clock in the morning. In addition it can be used to trigger certain actions when certain events occur.

3.2 Semantic Context-Aware Recommendations

We propose a context-aware thesaurus-based recommender system as a natural approach to semantically guide the query construction process.

In first place, we need a domain-specific thesaurus for recommendations, which must cover all relevant concepts of the domain and their relationships. We distinguish among three kind of relationships: broader, narrower and related. For example, "place" is a broader term for "restaurant" and "museum"; and "food" is related to "restaurant". Each concept also includes synonyms that may refer to the same idea, for example the concept "soccer" will also include the synonym "football". Moreover, these words can be specified in different languages without altering thesaurus structure, thus simplifying internationalization.

For our working prototype, we have tailored a thesaurus that covers concepts in the domain of transport, leisure and public services. In our opinion, they are the more likely to be useful in a generic-purpose mobile web search engine. Our thesaurus implementation has been defined in RDF [14] using the SKOS [7] vocabulary. It contains 100 concepts and about 200 relationships. As an example, figure 3 depicts one branch of our thesaurus.

The recommendation scenario works as follows. Once the user has typed one or more characters, the recommender system asks the thesaurus for concepts which start with those

Figure 4: Thesaurus-based suggestions tree.

Figure 5: Thesaurus concept and its ontological properties.

letters, of course including synonyms. Then it applies a rule-based filter to discard concepts which are not likely to be useful in that situation and to promote concepts that seem more interesting. The remaining concept list is sorted using a score system based on previous selections, and the best ones are selected to be shown. Instead of showing just the word for each concept, we also include its broader, related and narrower concepts in the same line, arranged in a tree view (as depicted in Figure 4). Finally, the user can select whichever option he deems more appropriate to fulfill his needs.

The first major feature of our proposal is context-awareness. Based on our context definition, we specified which are the best conditions for each thesaurus concept to be useful, and in which cases it could be directly discarded. These conditions are set by using SWRL rules, because the Context definition and the thesaurus can be joined together, and be seen as a unique ontology. This filter considerably reduces the number of options, whereas the remaining ones are the most interesting for a specific context.

A typical thesaurus rule matches the following pattern:

$$concept(?c) \wedge context(?t) \wedge \{conditions(?t)\} \rightarrow isSuitableInContext(?c, ?t)$$

Then, the *isSuitableInContext* property specifies which concepts are considered adequate in that specific context instance. Likewise, a property called *isNotSuitableInContext* asserts which concepts will probably be useless and should be discarded. The knowledge engineer is in charge of identifying under what conditions concepts are favorable or undesirable, based on his expert experience.

Our solution can also be used to solve sense ambiguity within queries. A common example of sense ambiguity is the word *jaguar*, which could refer to the animal, the car manufacturer or the Apple Operating System among others. Each of the senses has its own thesaurus concept, as they are homographs but represent completely different ideas. When there are two or more ambiguous options to recommend, the system offers all of them in separate lines, each one with its own relationships. In this case, the broader is really important to let the user distinguish every single meaning. For the *jaguar* example, the word will appear three times, one with *animal* as broader, another with *Car Manufacturer* and the last one with *Operating System*. When the user selects one, the system knows univocally to which one he refers in the thesaurus, and annotates the concept identifier to later expand the search engine query with disambiguating terms.

The last step of data acquisition is the use of the ontology to gather more information about user intentions and requirements. The ontology is designed to provide additional properties of the thesaurus concepts, together with their common values. Once the user selects a single concept, the system constructs a form to ask the user for more details, as shown in Figure 5. This can be used to construct a more concrete query which reduces ambiguity, but it is also a way of explicit user profiling, as the user is who specifies which options he prefer. In this case the user will be prompted to fill a small and query-specific form which takes just a few seconds and causes a direct effect on results. This beats other explicit solutions, which require users to fill big, generic, boring and time-consuming forms.

3.3 Personalized Search Engine

The semantic query construction assistant module is able to simplify user input task, but it also gathers a fair amount of extra information. We need to describe how to convert this information to a suitable format that will be processed on top of a traditional web search search engine.

There are two simple ways to employ all gathered information to affect search results. The first one is doing query expansion, given the selected thesaurus concept and the fields from the ontology, we can construct a new query by adding new terms to the user-typed query before accessing the traditional search engine. Another option is analyzing provided results after they are returned by the search engine, to filter out those which do not match user intentions. In this work we focus on the first one, query expansion.

The first consideration to take is that the user has a text field to type his own query in a traditional manner. Of course the system must be able to provide results even if any thesaurus concept or ontology option was selected. Therefore, the user query will have higher priority over the rest of the gathered information.

If the user selected one thesaurus concept we can profit that information to drive the search. For example we can add its synonyms and related concepts as optional terms, so web pages matching those subjects will gain weight in the search engine scoring system. This approach is interesting when a huge number of web pages is available, because it provides more specific results. On the contrary, when the amount of provided results is too scarce, we can try to launch a secondary query by substituting the main concept by one of the related ones. Of course this will result in a

performance and quality decrease, but this option is better than wasting user time and bandwidth usage to show a *No results found* webpage.

3.4 Enhanced User Interface

The heterogeneity of Mobile devices and browsers have led us to build a user interface that adapts to the capabilities of the target delivery context. For latest mobile browsers and devices, such as Windows Mobile PDAs or Apple's iPhone, we based it on HTML 4, AJAX and Javascript. For conventional mobile phones it has been degraded gracefully to XHTML-MP, providing a minimal but at the same time functional user experience. In every case, we tried to take full advantage of the screen and to minimize de number of clicks needed to navigate through the interface.

For enhanced web browsers we propose a tab-based search interface (see Figure 5): the first tab contains query terms and suggestions, the second one search results, and the last one allows the user to browse and modify some of the context properties, like current location, language, or situation (at work, at home, on a trip). When any of the search conditions change, the results tab is automatically updated to reflect those changes.

The search query tab contains an input box where the user can type his query in the traditional way. We wanted to maintain the traditional interaction flow to prevent the user from feeling lost. However, while he is typing, all the concepts (and related ones) that match the entered letters will appear on a tree view. Then he can select any of the suggested concepts (coming from the thesaurus), and finally additional options like concept properties (coming from the ontology) will appear (see Figure 5).

With regard to the implementation, it is important to note that we have taken advantage of the AJAX capabilities present in enhanced devices. For example, we minimize the traffic between the device and the server, making the user interface more responsive. We have also developed a browser plugin for the Windows Mobile Platform which provides access to Context information only accessible from the device, such as the GPS coordinates. Once they are obtained, the client properties are sent to the server, and they will be incorporated to the context definition as any other property.

4. EVALUATION

We invited 12 people, divided in 3 groups, to evaluate our prototype. The group A was composed by students in the first year of MsC in Computer Science; the second group, B, was integrated by IT technicians, and the third group, C, users not familiar with the mobile environment. We let them interact with our system using a real mobile device for a few minutes until they got used to it. Then we asked them to search for different kinds of information, including both subjects present and absent in the thesaurus. Finally they fulfilled a small survey containing questions regarding their user experience, particularly about the usefulness of the autocompletion system.

The most important point we wanted to test by using this evaluation was the level of satisfaction of final users. All members of our test groups found our suggestion system useful and intuitive, and they were able to use the system without trouble. They had the chance to directly compare the needed effort to type queries with our recommender system and without any aid tool. They agreed that the semantic recommender system reduces typing time and allows to use the system easier.

Less skilled people did not expect suggestions to appear, so they did not realize at the first time about the autocompletion feature. For that reason, we insist in the importance of maintaining the traditional search flow unmodified (type query, click on search button, browse results), while at the same adding the new features. When those users finally discovered the power of autocompletion they were the most amazed, as less skilled people normally needs even more time to accomplish a task in the absence of a recommendation system.

We also wanted to know their opinion about the multi-tab design applied to mobile web search, which permits to switch between the query/autocompletion page and the results page. They constructed the query with the autocompletion system, and then browsed the result page. When the results were not good enough to satisfy their information need, they naturally returned to the autocompletion tab to refine the query. They found easier to select another thesaurus concept or edit any of the ontology options rather than typing a new query from scratch. Indeed, users typically need several refinement iterations to reach their results, so our system let the users save even more time.

Our experiments revealed that the response time was longer than expected. In fact users got nervous and started switching tabs before waiting to load which resulted in even longer waits. For that reason we reduced load time by saving a local copy of each tab on the client side, minimizing the interactions with the server; now when the user is switching between tabs, and no option has changed, the cached content is used, therefore saving bandwidth and time.

5. CONCLUSIONS AND FUTURE WORK

We have described the limitations experienced by mobile web search users, focusing on lack of personalization and unconvenient input modes. We have developed a *semantic context-aware autocompletion* mechanism, based on a layered Context Model, a thesaurus to represent the subject domain, and an enhanced user interface. Our results show that these techniques provide a richer search experience.

We based our work on existing ideas from diverse research areas, like contextual applications, recommender systems and information retrieval query expansion. Then we studied how to join all of them together and construct a working prototype, which serves as proof of concept for the feasibility of the integration.

We have highlighted the importance of context modeling as the basis to provide personalization within mobile web search. Our proposed Context Model meets all the requirements imposed by a mobile web search environment. This is a first step towards a personalized access to the vast content of the World Wide Web. We proposed SWRL rules in order to extend de powerfulness of the context definition, allowing knowledge engineers to capture real world facts in a explicit manner.

We have observed that the thesaurus is a useful tool for several steps in the search process. It serves as guide for the recommender system providing the concepts of the domain, it is also useful to disambiguate user intentions and it can be easily annotated with rules to distinguish under what conditions each concept will be more or less interesting.

Our prototype is currently in an early phase and there is still a lot of research to do. We are working on integrating advanced user profiling algorithms to characterize user intentions in a more precise way, because knowledge about user is a fundamental piece in improving the relevancy of the results obtained during the recommendation process. We are also working on the personalized search module, analyzing different options of query expansion and result scoring to know how they improve search results.

6. ACKNOWLEDGMENTS

This work has been partially supported by TIN2006-15071-C03-02 project, Ministry of Education and Science, and by the Government of the region of Castilla y León through the Agency for Economic Development, (ADE), Spain.

7. ADDITIONAL AUTHORS

Additional authors are: Pablo de la Fuente (GRINBD, email: `pfuente@infor.uva.es`), Jorge Cabrero (GRINBD, email: `reybamba@gmail.com`), Guido García (ITDeusto, email: `ggarciab@itdeusto.com`), César Llamas (GRINBD, email: `cllamas@infor.uva.es`), and Álvaro Zubizarreta (GRINBD, email: `zubisoft@gmail.com`.

8. REFERENCES

[1] D. Brickley and L. Miller. FOAF vocabulary specification. Technical report, FOAF, nov 2007. http://xmlns.com/foaf/spec/.

[2] A. K. Dey. *Providing architectural support for building context-aware applications*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2000. Director-Gregory D. Abowd.

[3] P. Dourish. Seeking a foundation for context-aware computing. *Human-Computer Interaction*, 16(2/4):229–241, 2001.

[4] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: the concept revisited. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 406–414, New York, NY, USA, 2001. ACM.

[5] S. Hattori, T. Tezuka, and K. Tanaka. Context-aware query refinement for mobile web search. In *SAINT-W '07: Proceedings of the 2007 International Symposium on Applications and the Internet Workshops*, page 15, Washington, DC, USA, 2007. IEEE Computer Society.

[6] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML. W3C member submission, W3C, may 2004. http://www.w3.org/Submission/SWRL/.

[7] A. Isaac and E. Summers. SKOS simple knowledge organization system primer. W3C working draft, W3C, feb 2008. http://www.w3.org/TR/2008/WD-skos-primer-20080221/.

[8] Jo Rabin and Andrew Swainston. Content Transformation Landscape 1.0. W3C Working Draft, W3C, Oct. 2007. http://www.w3.org/TR/2007/WD-ct-landscape-20071025/.

[9] M. Kamvar and S. Baluja. Deciphering trends in mobile search. *Computer*, 40(8):58–62, 2007.

[10] M. Kamvar and S. Baluja. Query suggestions for mobile search: understanding usage patterns. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1013–1016, New York, NY, USA, 2008. ACM.

[11] K. Keenoy and M. Levene. Personalisation of web search. In *Intelligent Techniques for Web Personalization, IJCAI 2003 Workshop, ITWP 2003, Acapulco, Mexico*, pages 201–228, 2003.

[12] R. Kraft, F. Maghoul, and C. C. Chang. Y!q: contextual search at the point of inspiration. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 816–823, New York, NY, USA, 2005. ACM.

[13] S. Lawrence. Context in web search. *IEEE Data Engineering Bulletin*, 23(3):25–32, 2000.

[14] F. Manola and E. Miller. RDF primer. W3C recommendation, W3C, feb 2004. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.

[15] D. L. McGuinness and F. van Harmelen. OWL web ontology language overview. W3C recommendation, W3C, feb 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[16] Q. N. Nguyen and F. Ricci. User preferences initialization and integration in critique-based mobile recommender systems. In *Artificial Intelligence in Mobile Systems 2004, in conjunction with UbiComp 2004*, pages 71–78, Nottingham, UK, 2004.

[17] V. Roto, A. Popescu, A. Koivisto, and E. Vartiainen. Minimap: a web page visualization method for mobile phones. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 35–44, New York, NY, USA, 2006. ACM.

[18] A. Schmidt. A layered model for user context management with controlled aging and imperfection handling. In *Modeling and Retrieval of Context, Second International Workshop, MRC 2005, Edinburgh, UK*, pages 86–100, 2005.

[19] A. Sieg, B. Mobasher, and R. Burke. Ontological user profiles for representing context in web search. In *WI-IATW '07: Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, pages 91–94, Washington, DC, USA, 2007. IEEE Computer Society.

[20] R. T. A. Wietsma and F. Ricci. Product reviews in mobile decision aid systems. In E. Rukzio, J. Hkkil, M. Spasojevic, J. Mntyjrvi, and N. Ravi, editors, *PERMID*, pages 15–18. LMU Munich, 2005.

PHAROS – Personalizing Users' Experience in Audio-Visual Online Spaces

Ling Chen, Claudiu S. Firan, Wolfgang Nejdl, Raluca Paiu
L3S Research Center
Appelstr. 4
Hannover 30167, Germany
{chen, firan, nejdl, paiu}@L3S.de

ABSTRACT

The large volume of user generated content, although sometimes amateurish, represents a valuable source of information for audiovisual service providers. For example, companies and organizations can efficiently get feedback from consumers observing their online interaction with social media providers. Offering accurately personalized services is possible now, since users provide more personal information about themselves openly, which was previously much more difficult to perceive and measure.

In PHAROS, we aim at exploiting the new and freely available data to improve users' online experience with respect to their interaction with new media. We focus on building technologies, which bridge the gap between the availability of information (both in form of descriptions of content, such as annotations, and user interests and preferences) and the use of it, for augmenting traditional search and retrieval methods or for personalization purposes. In this paper, we describe how this external information can be brought into PHAROS and how it is used to support users, also describing the multiple components supporting this process.

1. INTRODUCTION

The amount of data available on the Web, in organizations and enterprises, is multiplying and data is increasingly becoming audiovisual. Search has become the default way of interacting with content and the ever-increasing data complexity leads to the necessity of a coherent approach to the growing variety of audiovisual formats, standards and tools. Users find themselves overwhelmed by the multitude of new audiovisual search tools, while businesses are at loss for stable direction. The growth of data volume is rapidly shifting to audiovisual content, yet the technologies that allow processing and retrieval of this content are either mainly experimental, or only vaguely capable of handling true queries and content. Audiovisual search is therefore one of the major challenges for organizations and businesses today, and search-based technologies which can provide contextually relevant, integrated and scalable access to distributed and heterogeneous collections of information are essential.

The PHAROS[1] European Integrated Project (Platform for searcHing of Audiovisual Resources across Online Spaces) aims at addressing these challenges and developing an innovative audiovisual technology platform, which takes user and search requirements as key design principles and is deeply integrated with user and context technologies. One of the objectives of the project focuses on the analysis, design and development of context and user technologies taking into account personalization and adaptability. This allows a social audio-visual interaction model to be integrated into the search engine, rather than using a traditional non-participatory information access model. PHAROS creates user interaction models where live user traffic continually improves the user experience via core primitives such as social network analysis or ranking based on trust.

The rest of the paper is organized as follows: In Section 2 we address the progress over the current state-of-the-art in different audiovisual search techniques focusing on user context. We continue in Section 3 with the description of the methodology used in PHAROS for personalization. Then, in Section 4 we present the architectural aspects of the PHAROS platform which support personalization. All modules are described in detail, also presenting the underlying algorithms. We finally conclude the paper in Section 5.

2. PROGRESS OVER STATE-OF-THE-ART

To achieve these ambitious objectives, PHAROS extends the state-of-the-art in the areas of core search technologies, as well as context and user technologies.

Regarding core search technologies, both XML search and content-based search are relevant. Previous work has addressed representation and semantic interoperability [6], as well as XML retrieval [4], [8]. Content based retrieval uses features of multimedia objects to facilitate their retrieval. [2], [10] focused exactly on this topic. However, emerging types of search patterns require both XML and content-based search to be integrated and made mature enough for industrial exploitation. PHAROS extends the state-of-the-art in this area by developing a scalable search platform with advanced query brokering to orchestrate audiovisual information access combining pluggable content-

[1] http://www.pharos-audiovisual-search.eu/

VLDB '08, August 24-30, 2008, Auckland, New Zealand.

based matching engines and schema agnostic XML based search kernels.

Context and user technologies have been tackled from various points of view: social media [9], [7], spam detection and ranking [3], [5] as well as security, trust and privacy [1]. PHAROS addresses all these aspects and more specifically focuses on exploiting user actions and interactions in personal and public spaces to provide advanced and semantically rich recommendations and personalized ranking algorithms. User and community profiles enable extreme precision for search, and exploit all kinds of user-generated metadata. Advanced spam detection algorithms, suitable for personalized ranking, are also provided and new lightweight forms of content protection are investigated.

3. PERSONALIZATION IN PHAROS

PHAROS is placed in a good position with respect to the new Web: there is a lot of momentum in users annotating and tagging audiovisual sources. However, there is a gap between the availability of this information and efficient exploitation for the purposes of improved access to desired content. Further, personalization has been known to suffer from the bootstrapping problem, where the experience of a new user can be unsatisfactory. In addition, there are other avenues of user information where users express their strong and personal opinion. This is the world of blogging – a very popular Web 2.0 phenomenon. Other public spaces including social networking sites and online forums are also rich sources of information about people and their preferences. The vision of PHAROS would be well served by creating technologies to bridge the aforementioned gap. For this purpose, in PHAROS, we currently take into account two types of important social media which is increasingly popular over the Web today: social annotations and weblogs.

Social annotation refers to the user-supplied tags, which are textual labels, to a piece of information on the Web, such as a picture, blog entry, a video clip etc. With the vast development of Web 2.0, social tagging has been a powerful and important feature provided by many social media applications, such as Flickr[2], Del.icio.us[3], and Last.fm[4]. Consequently, large volume of social annotation data can be collected easily, which enables reliable and accurate knowledge discovery. The knowledge embedded in such user-supplied annotation data is believed to be useful in many applications. Therefore, in PHAROS, we also investigate this: in particular, we focus on studying the usage patterns between users and social annotations. We then further explore the usage of discovered patterns in personalized search and recommendations.

Recently, weblogs have become one of the dominant forms of self-publication on the Internet. A weblog, or "blog", is commonly defined as a Web page with a set of dated entries, in reverse chronological order, maintained by its writer via a weblog publishing tool. The contents of entries (posts) are discussions and observations ranging from the mainstream to the very personal. The fast-growing popularity of the blogosphere offers new chances and challenges for Web search. For example, besides searching blogs, we can also analyze weblog communities, as a representative of our target audience, to predict the effectiveness of new recommendations. In PHAROS, by using weblogs we aim at discovering communities, which consist of blog users discussing similar topics in a certain period of time. We then analyze the properties of the identified communities, such as the information diffusion patterns in a community, to create accurate and detailed community profiles. The discovered community information is used to optimize the search and recommendation results for individual users.

4. SOCIAL MEDIA ARCHITECTURE

We start with the description of the architecture of social media modules in PHAROS, to show how the social data is brought to the PHAROS platform and exploited for personalization purposes. There are five modules in total which belong to three layers: ***Offline Analysis***, ***Storage*** and ***Processing***, as shown in Figure 1.

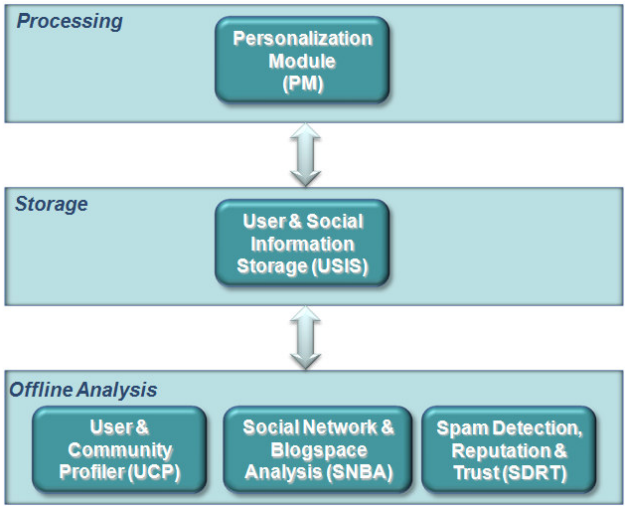

Figure 1. Architecture of Social Media Modules in PHAROS

There are three modules inside the Offline Analysis layer: *User & Community Profiler (UCP)*, *Social Networks & Blogspace Analysis (SNBA)*, and *Spam Detection, Reputation and Trust (SDRT)*. These modules retrieve related social metadata either from the PHAROS platform or from some other sources available on the Web. They further process the collected raw data to extract useful knowledge for other functionalities in PHAROS. In particular, the UCP module focuses on collecting and creating complete and accurate user profiles, as well as community profiles so that precise and personalized search and recommendation can be provided based on these profiles. The SNBA module aims at retrieving social network data, such as friendship network and blogspace information, and analyzing the social network both from a micro-perspective (e.g., the network of a user community) as well as a macro-perspective (e.g., the network of all PHAROS users). Due to the fact that current social media technologies are highly vulnerable to malicious users motivated by both private and commercial interests, the SDRT module is developed to improve the robustness of the PHAROS platform by detecting spam and assigning reputations and trust values to the users involved in social interactions. As SDRT is not implemented yet

[2] http://www.flickr.com/

[3] http://del.icio.us/

[4] http://www.last.fm/

inside the PHAROS platform we will focus only on UCP and SNBA as Offline Analysis modules.

The useful knowledge extracted from social metadata by all of the Offline Analysis modules is recorded in the same storage, the *User & Social Information Storage (USIS)*. Periodic updates are initiated in order to ensure that the stored knowledge is not obsolete. Beside the interfaces with Offline Analysis modules, USIS also interacts with the *Personalization Module (PM)*, located inside the Processing layer, to provide requested information for search and recommendation. The PM manifests the usefulness of social metadata in PHAROS. Various personalization strategies based on different extracted knowledge are developed to finally optimize the search results provided by PHAROS.

The details of each of the modules composing the Social Media Architecture will be described in depth in the following sections.

4.1 USIS – User & Social Information Storage

The *User & Social Information Storage (USIS)* plays a central role inside the architecture of the PHAROS platform, as this is the place where all user-related information is stored.

The functionality of the USIS is divided into the following roles:

- Metadata storage for PHAROS content objects;
- User related data storage and processing;
- External social interaction data storage.

Metadata Storage. Each PHAROS content object can have different types of metadata attached (e.g., tags, comments, ratings, and favorites). Each user having an account in the platform will be able to enrich content objects with metadata. This metadata can be viewed and searched by the users.

User Related Data. All the information stored in the USIS is meant to be later extracted and included into the personalization process. User and community profiling information, in form of both preferences as well as interaction with the PHAROS platform, are stored in here. Several parts build up the user profiles and are stored inside the USIS (details are presented in Section 4.2).

External Social Data. Data from external (not residing inside the PHAROS platform) sources like social networks or collaborative tagging Web sites can be also stored inside the USIS. This data can be used by any of the Offline Analysis modules in order to extract additional data relevant for PHAROS users or content objects.

4.1.1 Internal Structure of USIS

USIS is mainly a storage component, a database, providing several services depending on different data to be stored or requested. Currently, four major storage components reside inside the USIS.

Blog Analysis Storage. Blogs gathered from different sources are stored here. These blogs are further processed, and analyzed in detail by SNBA extracting interesting features and statistics needed by other components (e.g. UCP). The results of the analysis are stored in the Blog Analysis Storage as well.

Interaction Logs. User actions related to querying, receiving recommendations, and result handling is monitored by the PHAROS platform and are stored at the end of a user session in the USIS. Information stored here includes: what query was entered by the user, what results were viewed, what results were clicked, if the visualization of the results was interrupted prematurely (e.g. a video was not viewed until the end), etc.

User Generated Metadata. All user generated metadata resides in this storage component. This includes tags, comments, ratings and favorites. Information is stored as to which user added what metadata to which content object. In this way the metadata is filtered and statistics are computed regarding a user, a piece of metadata (e.g. some specific tag), or a content object.

User Profiles & Groups. User profiles, user-user relationships, and user-group memberships reside here. User profiles are constructed initially from the personal data a user enters when s/he creates his/her profile. UCP adds more data to the profile as the user starts using the platform.

4.2 UCP – User & Community Profiler

In order to achieve extreme precision in ranking and recommending multimedia content, adaptation of the core technology to user preferences and specific user contexts is necessary. Since most users may be unwilling to explicitly fill in and maintain a personal profile or they might not be able to specify an accurate profile, automatically inferring interests is important. Moreover, for providing high quality personalized services, profiles must be kept up-to-date as interests may change over time. Accurate user profiles often also depend on the community[5] a user belongs to. Therefore, inferring user profiles has to be complemented with the construction of community profiles. The *User & Community Profiler (UCP)* component is in charge of modeling user preferences from both inside and outside the PHAROS platform by collecting and automatically inferring information about users and communities they belong to. To overcome problems associated with modeling and finding communities adequately, it builds upon a model of user and community actions and interactions in social networks (provided by SNBA).

This module takes advantage of the explicit profile information freely provided by the PHAROS users and at the same time extends it with publicly available profile data from the services indicated by the user. Moreover, interests or preferences are implicitly found in concrete user (inter)actions and can thus be modeled from logging user interactions and group behavior within the PHAROS platform. Given this diverse amount of (raw) data about users and communities, the challenge and main focus of research activities within this module is to develop advanced techniques for building user and community profiles detailed, recent, accurate and reliable enough to meet the challenges of precise personalized and context-specific retrieval and recommendation.

In detail, the user or community profiles comprise:

- Explicit user information, given in the account/my profile section in the user interface, including basic data about users (gender, age, language) as well as some general interests;

[5] We use the term "community" when referring to any external social network of people e.g. build on platforms like Flickr; Social Group is used when we refer to the groups that users are actually building within the PHAROS platform

- User generated metadata (tags, comments or favorites) made within the PHAROS platform. Metadata in external Web 2.0 platforms like Last.fm, Del.icio.us or Flickr are analyzed and aggregated to infer preferences;
- Interests of individual user and communities extracted from the blogspace;
- Usage history. Issued queries and click through data from the interaction logs are used to show recently accessed resources ("charts") and serve as implicit feedback to infer likes and dislikes;
- Friendship or contact relations between users both inform about similarities or common interests and may be used for restricting recommendations based on privacy concerns;
- Similar users (neighbors) are automatically detected by combining data about users as described above.

Since users may have different preferences with respect to different situations, users can have multiple user profiles comprising the attributes listed – one for each of their various contexts (like "work", "leisure" etc.). For effectively supporting distinct user profiles, current active contexts have to be identified accurately to add the information to the right place. However, a default profile giving all information available is supported. To take into account most recent user and community data, profile updates are scheduled according to availability of new data.

4.2.1 Functionalities of UCP

At the current stage, the UCP module performs the analysis as follows.

Log Analysis. What resources a user searches for, which multimedia resources he actually accesses (for how long), whether he even recommends them to other people, as well as which persons he frequently interacts with, provide a lot of valuable data about a user's topics or preferences and probably typical behavioral patterns. In contrast to explicitly provided profile information, such implicit feedback to resources and people has to be analyzed to infer meaningful and generalizable user attributes to optimize personalization. For example, the user evaluation of the result set (skipped and clicked items) helps to infer new associated terms by getting keywords from resources implicitly judged as relevant. On the other hand, similar resources listened to or watched can be exploited to find similar queries or even super-ordinate topics. In general, just building the list of (recently) seen items - usage history of a user – alone is very useful for profiling interests with respect to finding similar users or similar resources. Interactions within groups and with friends or unknown people are analyzed to model the social network of a user which can again be used to infer commonalities and preferences. Other patterns to be mined from action sequences (like system internal navigation paths) help personalizing view settings and creating navigational short cuts as well as to inform about general usability issues to be improved. All these extracted and inferred information are translated into specific attributes and written to the user profile.

Annotation Analysis. By adding tags, comments or favorites to multimedia resources seen within PHAROS, users tell us (implicitly) about what they like, don't like or what topics they are interested in, as they organize their resources around it. Therefore, Annotation Analysis gathers this kind of data and analyzes it in depth to make it available for profiling and search.

Opinion Mining. For comments, stopword removal and term normalization are standard procedures, important keyword extraction and Sentiment Analysis / Opinion Mining are more advanced techniques. This aims at analyzing any free textual annotations about content objects within PHAROS or elsewhere on the Internet. From these textual annotations new tastes about audiovisual objects are deduced, and the user profile is updated with this new knowledge.

Tag Analysis. Tag analysis comprises first of all normalization and the inclusion of alternative, synonymous labels. Also absolute and relative frequency information is calculated for individual tags as well as co-occurrence relationships that may help to dissolve term ambiguities or to refine queries by synonyms / strongly associated terms. To exploit the potentially huge effort a user already invested in tagging interesting Web pages, songs or pictures in one or the other Web 2.0 platforms, tag analysis also fetches and analyzes the user's tags from external sites like Last.fm and Del.icio.us.

Profile Building. Finally, the results of the above mentioned single analyses have to be merged and enriched with the information explicitly given by the user. Both types of information go directly into the profile under the corresponding attributes. Note that this may mean merging or resolving conflicts about preferred topics identified in the single analyses.

4.3 SNBA – Social Networks & Blogspace Analysis

The *Social Networks & Blogspace Analysis (SNBA)* module aims at gathering and analyzing social network data coming from blogs or friendship networks for discovering additional knowledge which can be applied to improve the search and recommendation results in the PHAROS platform.

There are many different types of blogs, differing not only in the type of content, but also in the way that content is delivered or written. However, for our analysis we focus on personal blogs, as this type of blogs – on-going diaries or commentaries by individuals – reveal the most personal information about their authors. Since all blogs are on the internet by definition, they may be seen as interconnected and socially networked. Several features permit bloggers to link to each other's blog pages: the so-called "blogrolls" lists one's favorite blog list in a frame inside their own blog page. These links represent other authors' blog pages that this author considers interesting and frequently visits for reading and / or directly commenting. In a sense this feature is similar to the in-links of a Web page: they inject some importance to the blog pages they target by the fact that the author of the blog page lists these links on his own and indirectly shows that there are some trusted blog sources, worth reading. Besides, the blogrolling phenomenon is somewhat reciprocal. By linking to a blog, users are increasing their blog's chances of being linked-to by other weblogs. These links between Weblogs are the "currency" of the Weblog community. The more links one has pointing to his weblog, the more likely he gets a growing audience and high rankings in search engines. A blogroll helps a user get started earning links from other weblogs by expressing affiliations. Permalinks are also a possibility to create social links among bloggers. Unlike blogrolls which point to a blog page, a permalink represents a link to a particular blog post inside a blog page (created by the author of this page) and allows other bloggers to

use it to jump directly to this blog entry. Given the highly dynamic content change of the blog pages, this feature is extremely useful if users want to re-read some very interesting post, which has already passed from the front page to the archives.

Given the aforementioned characteristics of the weblogs and of the blogosphere, it can be easily seen that blogging is inherently a social process, one in which information is created and diffuses (or flows), between bloggers due to bloggers influencing and being influenced by other bloggers. Consequently, the overall objectives of the SNBA module are to:

- Determine ways to capture what is diffusing;
- Determine paths for who influences whom;
- Measure the extent of this influence;
- Exploit this knowledge for personalized ranking and search.

We touch each of the objectives presented above, describing the components and algorithms for exploiting information diffusion for personalized ranking and search.

4.3.1 Functionalities of SNBA

Several components build up the SNBA module, such as Blog Identification and Blog Ingestion etc. We focus on describing the main component Blog Analysis Processing.

The Blog Analysis Processing component is further divided into different sub-modules (see Figure 2). It works as a three-stage process: at the lowest level, a static snapshot representation of the domain is obtained through the Text Mining Module. Based on this, higher level abstractions are built (Topic / Community Detection). Then, dynamics and evolution are handled within the Information Diffusion Module. Finally, a Profile Extraction component aggregate discovered knowledge into users' as well as communities' profiles.

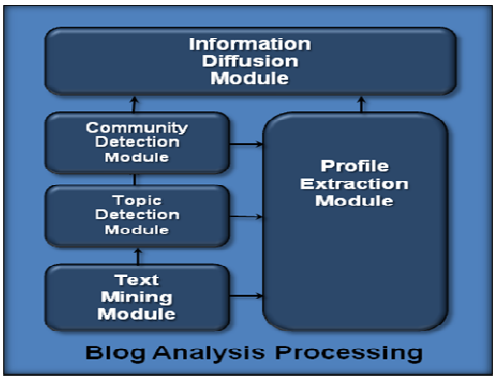

Figure 2. Blog Analysis Processing

The Text Mining Module uses pre-processed blog posts as input for the mining process. The output (typically described by a set a probability word pairs) is used as input for determining the topic of blog posts or of the blogs themselves. Community structures are then created from these underlying descriptions. Topics from blogs and posts written by a user are taken as representations for the user's and her communities' profiles. Once communities have been detected – the dynamics and evolution of information between individuals and communities are mined, in the Information Diffusion Module. Finally, the Profile Extraction Module updates the created profiles with topic, time and information diffusion information etc. Due the space constraints, we ignore the discussion of the algorithms employed by each module. Interested readers can refer to [21] for the detail.

4.4 PM – Personalization Module

The *Personalization Module (PM)* focuses on providing personalized search and recommendation functionality. This component is especially important because it unlocks the value of personal information stored in USIS in order to improve the users' experiences inside the PHAROS platform. PM takes requests from the user and uses information stored in USIS as basis for performing personalization. The relevant information about user interests is computed offline in the UCP and SNBA modules and is then retrieved by PM both during the pre-computation of personalized ranking values, as well as during the model building phase of the recommendation engine. This model is later used to compute recommendations online.

4.4.1 Internal Structure of PM

The component architecture has been designed in order to support *pluggable* recommendation algorithms (see Figure 3), which can be further developed and extended, for example to adapt the behavior of the PM module depending on the context or the user data available, and to try to complement some weaknesses and strengths of the algorithms themselves, by creating hybrid models that combine them.

In case of the Personalized Search Component there are two critical factors for the effective personalization: the quality of the user profile and the query processing time. The user profiles are pre-computed by UCP and SNBA components and stored in the USIS module. The PM module communicates with USIS to fetch and transform these profiles into a format required by different personalization algorithms. During query time, the system sends a request with the original query to PM and receives back a new query which includes the necessary modifications for a better ranking.

One part of the PM module is the *Query Personalization Component* (see Figure 3). When a query comes from the system via the provided API, the *Query Parser* transforms it into internal format for further processing. The *Personalization Selector* component chooses the requested personalization method and asks the *Profile Retriever* for a necessary user or community profile information. The *Query Personalizer* transforms the original query and sends the resulted personalized query back to the system for retrieval.

The *Recommender System Component* also depends directly on the user profiles pre-computed by UCP and SNBA. Once these profiles are retrieved from USIS, the *Modeler* sub-component builds a model of the user preferences. The computation is done offline periodically and the results are also stored back into the USIS. Once the model has been built, the Recommendation Engine is ready to compute the necessary list of personalized recommendations for a given user, as well as her neighborhood (User-based recommendations). Depending on the context, the Recommendation Engine is also capable to recommend similar items given a resource (Item-based recommendations).

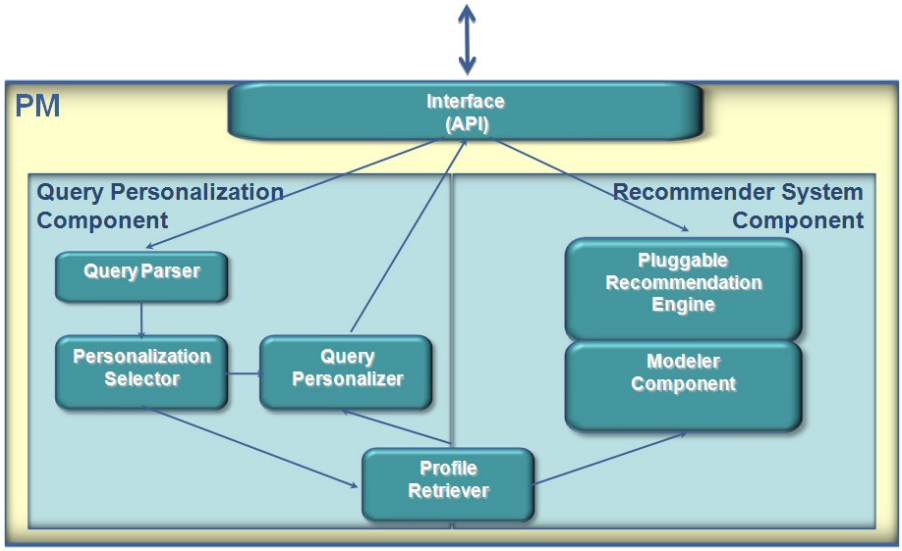

Figure 3. Internal Structure of the Personalization Module

The *Query Personalization* and *Recommender System* are the fundamental components of PM and below we describe each of them in detail.

4.4.2 Query Personalization Component

In addition to general search capabilities, PHAROS provides also personalized search results matching the profile of the user or any groups the user belongs to. Personalization involves both a filtering and ranking of results. Result filtering is used to limit the result set to content, which fits the user's information need, and content the user has permission to view. Ranking of results takes into account user and group preferences and ranks content, believed to be of high relevance to the user, higher than content which is of general interest. Ranking parameters are part of the query, and are inserted by the PM module. Different personalization techniques are developed within Information Retrieval field, like query re-weighting and query expansion, just to name a few. We provide details on implemented methods in Section 4.4.3.

The personalized search capabilities assume re-ranking of the relevant multimedia items using information about previous user's interactions with the PHAROS platform. All historic usage data, such as user's queries, clicked results, tags in use, etc., should be exploited to provide a more precise search output. For providing high quality personalized services, user profiles must be kept up-to-date as interests may change over time.

4.4.3 Personalized Search Algorithms

For the *Query Personalization Component*, we implemented 6 Information Retrieval algorithms for Relevance Feedback and Query Expansion. The algorithms can be used as standalone methods as well as in combination with each other. The effectiveness of the proposed methods has been proved in general text and multimedia retrieval, while their practical usefulness depends on available data and quality of the user profiles:

(1) Fields Reweighting. Results are initially ranked using default values for the given query keywords – fields. Based on previously collected information regarding user tags and associated ratings an algorithm can specify a different weight for each field (query keyword) and this information is then used for ranking. The frequently used user's tags and query fields receive higher weights and results are biased towards them. This relevance feedback technique is based on a well-known Rocchio method [11]. Long-term feedback is obtained from tag usage of the user or user group and is captured from UCP module in the form of a user profile, which tracks user-specific weights and other feedback-based parameters. The vector-space representation of the query is modified so that more important term dimensions are emphasized and similarity between query and each item of interest is affected. Top-N most similar items are then presented to the user.

(2) Results Filtering. Create restrictions based on the user profile, like removing from the ranked list of results the items that the user dislikes. We use Generalized Query Point Movement method [12], where previously received poor ratings from a user are used to extract tags representing what the user does not like. The result items containing such tags are moved down the ranking. The algorithm has a similar mechanism to Fields Reweighting technique, but negative assessments are used to compute the user profiles. This personalization technique is effective when users explicitly mark items as uninteresting.

(3) Query Expansion. Based on the users' tag usage patterns we compute tags' similarity to each other. Additional keywords are added to the query based on preferences from the user profile. This method [13] is one the most frequently used for personalization and can significantly increase recall in situations, where original query does not have enough results. Query expansion is essentially adding new features to the query vector and re-ranking the results accordingly. The initial query terms are still of higher importance for the ranking. The precise values for the algorithm tuning have to be defined based on available data, which can be done as soon as user profiles and interaction histories are collected.

(4) Query by Example. For the scenarios when a user wants to find items similar to a current one we consider tags similarity. A tag description of the selected example is used as a query and a set

of results, excluding original item, is returned to a user. The personalization part of this method assumes additional query modification, based on recent user queries and tags used. A method ranks higher the results which are not only similar to a given example, but also remind previously seen items.

(5) Community-Tag Rank. This algorithm is inspired by work on Topic-Sensitive PageRank [14] and [15], but based on textual similarity rather than link-based similarity. Document model includes all the fields that can be extracted for multimedia content. We create a community-tag vector for each of the identified communities. To make computation scalable we assume that number of communities is significantly lower than a total number of users. A Community-Tag Rank represents a similarity between an item and a community vector, which is composed from tag usage statistics of all users belonging to the community. Communities can be defined based on explicit membership in particular communities and automatically computed clusters of users. A user belongs to one or more communities (topic groups) and we can compute a linear combination of the community vectors for items before query time, which is called Community-Tag Rank. A single score of Community-Tag Rank is associated with each multimedia item—community pair. During the computation of the item-query similarity the personalized Object Rank vector is used as a factor for ranking as a query-independent parameter.

(6) Community-Rating Rank. This method is similar to Community-Tag Rank, but is based on a collaborative filtering rather than document model. A community profile for a Community-Rating Rank computation consists of previously issued users' ratings and independent of multimedia item tags. This average rating allows re-ranking retrieved items with respect to their overall popularity among community members and quality of each returned result. As with Community-Tag Rank, this value is query independent and it is pre-computed. During query time this value is added to the item relevance score.

4.4.4 Recommender System Component

Recommender Systems support people by identifying products or services they appreciate, helping them to face the information explosion, where the complexity of offers exceeds the user's capability to survey them and reach an optimal decision.

Different approaches have been suggested for supplying meaningful recommendations to users and some of them implemented and deployed successfully over e-commerce and services sites like Amazon[6], Netflix[7], or MyStrands[8].

State-of-the-art Recommender Systems mostly use a variant of Collaborative Filtering (CF), an approach to solve the recommendation task that relies on historical data gathered from users, rather than using the information about content. The underlying assumption of the CF approach is that those who agreed in the past tend to agree again in the future, capturing human behavior: people searching for an interesting item of which they have little or no information, tend to rely on friends to recommend items they tried and liked.

[6] http://www.amazon.com/

[7] http://www.netflix.com/

[8] http://www.mystrands.com/

The goal of the PM's *Recommender System Component* is to identify neighborhoods of users with similar taste, based on the profiles built by the UCP and SNBA modules and stored in the USIS. To build a user's neighborhood, the Recommender System Component relies on information of past user interactions (e.g., explicit ratings, tags assigned) or implicit grading methods based on user behavior actions, such as the time spent on a particular item Web page. In order to provide recommendations for a given user, the system uses her corresponding neighborhood to compute a list of items interesting for her. A similar approach is also taken to consider neighborhoods of similar items to be exploited in order to provide recommendations of similar contents and resources.

4.4.5 Recommendation Algorithms

In the case of the *Recommender System Component* the following algorithms are provided, and are also combined with each other to provide the target functionality:

(1) Tag-aware Collaborative Filtering. It exploits the tag-based profiles, in both dimensions (user, tag) and (item, tag) to build user and item neighborhoods in order to compute personalized recommendations [16]. Tag-based user profiles are defined as collections of tags together with corresponding scores representing the user's interest in each of these tags. Once the profiles have been computed, they are arranged in a User-Tag matrix structure, which is then used to derive the recommendations applying CF techniques that group similar users in order to suggest them valuable items that in turn have been inferred by their associated tags.

(2) Standard User-based Collaborative Filtering. It supports (1) and can also be used alone or as part of other Recommendation Engine to exploit different kinds of profiles, and not only explicit ratings as in traditional CF [17], [18]. The recommendations for each individual user are obtained by identifying a neighborhood of similar users and recommending items that this group of users found interesting. The design recommendations described by [19] have been also considered in the implementation of this algorithm.

(3) Standard Item-based Collaborative Filtering. The recommendation task in this case is focused on the items' similarity, rather than on the users' similarity. It also supports (1) and its main objective is to produce a list of recommendations given a target item [20]. This recommendation algorithm uses the item-to-item similarities to compute the relations between the different items. It builds a model that captures these relations and then applies this model to derive the top-N recommendations for an active user. The model, which at the core is an item-item matrix representation, is built based on the original user-item matrix of user profiles that reflects their aggregated historical information of consumed items. Each item is associated with a vector in the users' space, and these vectors are then used to compute the similarity among the items. Once the similarities have been computed, for each item, just the most similar k items are kept on the model, where k is an input for the algorithm. The model computed is used during the recommendation step, where the goal is to recommend similar items for a given one.

5. CONCLUSIONS & FUTURE WORK

In this paper, we focus on describing social media metadata based personalization supported in PHAROS. The vision for PHAROS

has an important place for user generated social metadata; the use of personalization and recommendation is vital to PHAROS for enhancing the user experience. The content based search service is served better by an accurate and efficient social based search service. Consequently, social media data analysis and processing plays an important role in audiovisual online spaces.

Particularly, we describe four social media related modules developed in this project. Two analysis modules, User & Community Profiler (UCP) and Social Networks & Blogspace Analysis (SNBA), aim at performing analytic study on various user-generated social media data and extracting knowledge relevant to users' interests. This extracted knowledge is further exploited by the processing module, Personalization Module (PM), to enhance users' personal search experience. A data storage module, User & Social Information Storage (USIS), is also provided to accommodate not only raw social media data but also processed and extracted information from the raw data.

There are still a few open issues in successful exploring social media data for personalization within PHAROS, such as scalability and robustness. Our ongoing work include improving the algorithms employed by each module to address these issues, as well as conducting more research and developing work in optimizing the functionalities provided by social media modules by large scale.

6. ACKNOWLEDGMENTS

This work was partially supported by the PHAROS project funded by the European Commission under the 6th Framework Programme (IST Contract No. 045035).

7. REFERENCES

[1] Aichroth P., Puchta S., Hasselbach J. Personalized Previews: An Alternative Concept of Virtual Goods Marketing, *Virtual Goods Conference, 2004.*

[2] Aslam J., Montague M. Models for Metasearch. *SIGIR, 2001.*

[3] Bharat K., Henzinger M. R. Improved algorithms for topic distillation in a hyperlinked environment, *SIGIR, 1998.*

[4] Carmel D., Maarek Y. S., Mandelbrod M., Mass Y., Soffer A. Searching XML Documents via XML Fragments. *SIGIR 2003.*

[5] Carvalho A., Chirita P. A., Silva de Moura E., Calado P., Nejdl W. Site Level Noise Removal for Search Engines. *WWW, 2006.*

[6] Dong X., Halevy A. Malleable Schemas. *WebDB, 2005.*

[7] Ghita S., Nejdl W., Paiu R. Semantically Rich Recommendations in Social Networks for Sharing, Exchanging and Ranking Semantic Context. *ISWC 2005.*

[8] Kakade V., Raghavan P. Encoding XML in Vector Spaces. *ECIR, 2005.*

[9] Kumar R., Novak J., Raghavan P., Tomkins A. On the bursty evolution of blogspace. *WWW, 2003.*

[10] McDonald K., Smeaton A. A Comparison of Score, Rank and Probability-based Fusion Methods for Video Shot Retrieval. *CIVR, 2005.*

[11] Rocchio, J. J. Relevance feedback in information retrieval. *Prentice-Hall, 1971.*

[12] Ortega-Binderberger, M., & Mehrotra, S. Relevance Feedback in Multimedia Databases. *In Handbook of Video Databases: Design and Applications, 2003.*

[13] Qiu, Y., & Frei, H. Concept Based Query Expansion. *SIGIR, Pittsburgh, 1993.*

[14] Haveliwala, T. H. Topic-sensitive PageRank. *Proceedings of the Eleventh International World Wide Web Conference, Honolulu, 2002.*

[15] Chirita, P. A., Nejdl, W., Paiu, R., & Kohlschütter, C. Using ODP Metadata to Personalize Search. *Proceedings of the 28th ACM International SIGIR Conference on Research and Development in Information Retrieval, Salvador, 2005.*

[16] Firan, C., Nejdl, W., & Paiu, R. The Benefit of Using Tag-Based Profiles. *Proceedings of the 2007 Latin American Web Conference, Santiago, 2007.*

[17] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *Proceedings of Conference on Computer Supported Coorperative Work , 1994*

[18] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM, 1997.*

[19] Herlocker, J. L., Konstan, J. A., & Riedl, J. An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. *Information Retrieval, 2002.*

[20] Deshpande, M., & Karypis, G. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems, 2004.*

[21] A.Stewart, L. Chen, R. Paiu & W. Nejdl. Discovering Information Diffusion Paths from Blogosphere for Online Advertising. *ADKDD, California, 2007.*

Capturing Multiple Interests in News Video Retrieval by Incorporating the Ostensive Model

Frank Hopfgartner, David Hannah, Nicholas Gildea, Joemon M. Jose
University of Glasgow
Glasgow, United Kingdom
{hopfgarf, hannahd, ngildea, jj}@dcs.gla.ac.uk

ABSTRACT

We propose an adaptive news video retrieval approach which is based on the Ostensive Model of developing information needs. We therefore introduce a news video retrieval system called NewsBoy which captures the users' implicit interactions with its graphical interface, extracts terms from visited video documents and stores them in user profiles. The terms are weighted based on the type of implicit feedback, multiple interests are identified by clustering the content of the profile. In this paper, we describe the architecture of the system and introduce our approach of adding the ostensive factor to capture the users' evolving interest. Preliminary results show the acceptance of the system and highlights drawbacks.

1. INTRODUCTION

Consuming information has a central impact on the development of our society, leading to the transformation from the industrial to the information age. Newspapers, television news broadcasts, the WWW and other sources provide the society with a vast amount of information, an increasing percentage of which is in digital format. However, facing this excessive supply of information sources might overwhelm information consumers. Hence, there is a need to provide personalised access to them.

Arezki et al. [1] provide an example to explain the need of a personalisation service: When a computer scientist enters the search query "java" into a search engine, he is most likely interested in finding information about the programming language. Other people, however, might expect results referring to the island of Java in Indonesia or a type of coffee beans bearing this name. Sebe and Tian [18] discuss that for providing personalised information based on multimedia content, sophisticated research in various areas is needed, including the acquisition of user preferences and how to filter information by exploiting the user's profile.

A classical approach to capture the user's preferences is profiling. User profiles can be used to create a simplified model of the user which represents his interests on general topics. Commercial search engines incorporate such profiles, the most prominent being Google offering iGoogle and Yahoo! offering MyYahoo!. Query expansion is used to gather the user's interest and search results are re-ranked to match their interests.

These services rely on users' explicitly specifying preferences, a common approach in the text retrieval domain. By giving explicit feedback, users are forced to update their need, which can be problematic when their information need is vague [21]. Furthermore, users tend to provide not enough feedback on which to base an adaptive retrieval algorithm [8]. Deviating from the method of explicitly asking the user to rate the relevance of retrieval results, the use of implicit feedback techniques helps by learning user interests unobtrusively. The main advantage is that users are relieved from providing feedback. A disadvantage is that information gathered using implicit techniques are less accurate than information based on explicit feedback [14].

A challenging problem in user profiling is the users' evolving focus of interest. What a user finds interesting on day A might be completely uninteresting on day B, or even on the same day. The following example illustrates the problem: Joe Bloggs is rarely interested in sports. Thus, during Euro 2008, the European Football Championship, he is fascinated by the euphoria exuded by the tournament and follows all reports related to the event. After the cup final, however, his interest slowly abates again. How to capture and represent this dynamic user interest is an unsolved problem. Moreover, a user can be interested in multiple topics, which might evolve over time. Instead of being interested in only one topic at one time, users can search for various independent topics such as politics or sports, followed by entertainment or business.

In this paper, we introduce NewsBoy, a personalised multimedia application which is designed to capture the user's evolving interest in multiple aspects of news stories. NewsBoy automatically processes the daily BBC One news bulletin and recommends news stories by unobtrusively profiling the user based on his interactions with the system. The news aspects are identified by clustering the content of the profile. We introduce four different functions that incorporate the evolving interest of the user and evaluate the effect of these functions on the profiles.

The paper is structured as follows: Section 2 provides an overview of related work. Section 3 introduces the architecture of NewsBoy. In Section 4, we introduce our approach of capturing the user's interactions by extracting relevant

VLDB '08, August 24-30, 2008, Auckland, New Zealand

terms from results a user interacted with, combining them with a relevance weighting and storing them in a user profile. In order to capture the user's evolving interest, we adopt the ostensive model to manipulate the weighting of the terms in accordance to the iteration when they were added to the profile. Further, we introduce our methodology of clustering these terms to represent the user's multiple interests in different aspects and present a preliminary evaluation in Section 5. Finally, we discuss the system in Section 6.

2. BACKGROUND

Our work builds on a number of research areas, including news video retrieval, personalised news delivery and techniques to capture evolving user needs. In the following, we introduce the state-of-the-art of these areas.

2.1 News Video Retrieval

Nowadays, almost every television channel has its own news bulletin, indicating that television is a widely accepted mass media to provide consumers with the latest news. Consequently, processing television news has been an important research area and much recent work, such as that represented by the TRECVID [20] research effort, aims to tackle the difficult problems of content based video retrieval. While some systems have a particular emphasis on the system side, other research efforts are looking towards improving state-of-the-art video retrieval techniques from the user's point of view, such as the Open Video Project[1].

A number of conclusions can be drawn from these efforts: first of all, video retrieval is not as sophisticated as its textual counter part. The reason for this is the so-called "semantic gap" [10], the difference between the low-level representation of video and audio data, and the high-level semantics which the user would ideally like to associate with retrieved data. Furthermore, segmenting and indexing video is a challenge. Considering a news broadcast as a unit of retrieval will generate a result list containing whole video documents. A user must watch or browse through the whole video to finally find the information he wants, a demanding approach. Hence, it is necessary to split videos into smaller, semantically related, *segments* which should ease the access of the video data. In text retrieval, techniques have been developed to identify relevant sections of the text, e.g. [17] and to segment documents based on these sections. Hence, users can easily browse through short results to satisfy their information need. Boreczky et al. [3] argue that television news consists of a collection of *story units* which represent the different events being relevant for the day of the broadcast. An example story unit from the broadcasting news domain is a report on yesterday's football match, followed by another story unit about the weather forecast.

Indexing these segments, i.e. based on textual annotations or visual representations of the segments provides an easy access to the data collection. A challenging approach however is to identify these stories a user is really interested in. The problem will be introduced in the following section.

2.2 Personalised News Delivery

Web 2.0 facilities enable everyone to easily create their own content and to publish it online. Users can upload videos on platforms such as YouTube, share pictures on Flickr or publish anything in a weblog. Two direct consequences of this development can be identified: first of all, it leads to a growing quantity of content presented in a multimedia format. Secondly, information sources are completely unstructured and finding interesting content can be an overwhelming task. Hence, there is a need to understand the user's interest and to customise information accordingly.

A common approach to capture and to represent these interests is user profiling. Using user profiles to create personalised online newspapers has been studied for a long time.

Chen and Sycara [6] join internet users during their information seeking task and explicitly ask them to judge the relevance of the pages they visit. Exploiting the created user profile of interest, they generate a personalised newspaper containing daily news. However, providing explicit relevance feedback is a demanding task and users tend not to provide much feedback [8].

Bharat et al. [2] create a personalised online newspaper by unobtrusively observing the user's web-browsing behaviour. Although their system is a promising approach to release the user from providing feedback, their main research focus is on developing user interface aspects, ignoring the sophisticated retrieval issues.

Smeaton et al. [19] introduced Físchlár-News, a news video recommendation system that captured the daily evening news from the national broadcaster's main TV channel. The web-based interface of their system provides a facility to retrieve news stories and recommends stories to the user based on his interest. According to Lee et al. [13], the recommendation of Físchlár-News is based on personal and collaborative explicit relevance feedback. The use of implicit relevance feedback as input has not been incorporated.

Profiling and capturing the users is an important steps towards adapting systems to the user's evolving information need. In the following section, we introduce the problem of capturing this evolving need.

2.3 Evolving User Needs

In a retrieval context, profiles can be used to contextualise the user's search queries within their interests and to re-rank retrieval results. This approach is based on the assumption that the user's information interest is static, which is however, not appropriate in a retrieval context.

Campbell [4] argues that the users' information need can change within different retrieval sessions and sometimes even within the same session. He states that the user's search direction is directly influenced by the documents retrieved. The following example explains this observation: Imagine a user who is interested in red cars and uses an image retrieval system to find pictures showing such cars. His first search query returns him several images including pictures of red Ferraris. Looking at these pictures, he wants to find more Ferraris and adapts the search query accordingly. The new result list now consists of pictures showing red and green Ferraris. Fascinated by the rare colour for this type of car, he again re-formulates the search query to find more green Ferraris. Within one session, the user's information need evolved from red cars to green Ferraris. Based on this observation, Campbell and van Rijsbergen [5] introduce the ostensive model which incorporates this change of interest by considering *when* a user provided relevance feedback. In the ostensive model, providing feedback on a document is seen as ostensive evidence that this document is relevant for

[1] `http://www.open-video.org`

the user's current interest. The combination of this feedback over several search iterations provides ostensive evidence about the user's changing interest.

There are different types of interaction feedback, usually divided into two categories: explicit and implicit feedback. Explicit feedback is given when a user actively informs a system what it has to do on purpose, such as selecting something or marking it as relevant. Implicit feedback is given unconsciously. An example is printing out a web page, which may indicate an interest in that web page. The basic assumption is that during a search, users' actions are used to maximise the retrieval of relevant information. Implicit indicators have been used and analysed in other domains, such as the WWW [7] and text retrieval [12], but rarely in the multimedia domain. However, traditional issues of implicit feedback can be addressed in video retrieval since digital video libraries facilitate more interactions and are hence amenable to implicit feedback.

This section introduced the research domains of our work and argued about the research problem of capturing the evolving user need in order to personalise news videos in accordance to the user's interest in multiple aspects of the news. In the next section, we introduce NewsBoy, a news video retrieval system which incorporates the previously introduced research domains.

3. NEWSBOY ARCHITECTURE

NewsBoy is a web based news video retrieval system based on AJAX technology. AJAX takes away the burden of installing additional software on each client (assuming that JavaScript is activated and a Flash Player running on the client side).

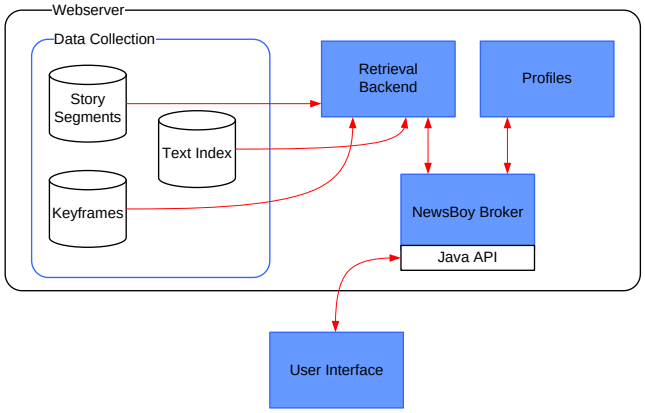

Figure 1: NewsBoy Architecture

Figure 1 illustrates the conceptual design of the system. As the graphic shows, NewsBoy can be divided into five main components, four running on a web server and one, the user interface, on the client side. The first component is the data collection which will be introduced in Section 3.1. The retrieval backend, the second component of NewsBoy, administers the data collection. We are using MG4J[2], an open source full-text search engine. The third component is the user interface, which runs on the client side. It will be introduced in Section 3.2.

[2]`http://mg4j.dsi.unimi.it/`

3.1 Data Collection

In the scope of this research, we focus on the regional version of the BBC One O'Clock news. The programme covers international, national (UK) and regional (Scotland) topics, which are usually presented by a single newsreader. The bulletin has a running time of 30 minutes and is broadcasted every day from Monday till Friday on BBC One, the nation's main broadcasting station. The BBC enriches its television broadcast with Ceefax, a closed caption (teletext) signal which provides televisual subtitles for the deaf. The data collection we used for this study consists of 115 editions of the daily news broadcast which have been recorded constantly over the past few months. Based on its textual, visual and audio features, we segmented the news videos into semantically related story segments, the unit of retrieval in our system. The index contains 2963 stories, which are aligned with 4.1 non-stopword-terms on average.

During the period of the recording, various main events have been dominant in the news. In relation to the evaluation date of this study (April 2008), these events can be classified into (1) latest, (2) recent and (3) past events. Here, we give some examples:

1. Latest events (current week): Discussions about air travel.

2. Recent events (2 month ago): Reports about the International Bank Crisis.

3. past events (>4 month ago): Christmas time

3.2 Interface

Figure 2 shows a screenshot of the NewsBoy interface, its features will be described in the following section. The interface can be divided into three main panels, search panel (A), result panel (B) and clustered search queries (C).

In the search panel (A) users can formulate and carry out their searches by entering a search query and clicking the button to start the search. BM25 [16] is used to rank the retrieved documents in accordance to their relevance to a given search query.

Once a user logs in, NewsBoy displays the latest news stories in the result panel (B). Moreover, this panel lists retrieval results. The panel displays a maximum of 15 results, further results can be displayed by clicking the annotated page number (1). The results can be sorted in accordance to their relevance to the query or chronologically by their broadcasting date (2). Results are presented by one keyframe and a shortened part of the text transcript. A user can get additional information about the result by clicking on either the text or the keyframe. This will expand the result and present additional information including the full text transcript, broadcasting date, time and channel and a list of extracted named entities[3] such as persons, locations and relative times (3). In the example screenshot, the second search result has been expanded. The shots forming the news story are represented by animated keyframes of each shot. Users can browse through these animations by clicking on the keyframe. This action will center the selected keyframe and surround it by its neighboured keyframes. The keyframes are displayed in a fish-eye view (4), meaning that

[3]We use the General Architecture for Text Engineering (`http://gate.ac.uk`) for the extraction of named entities.

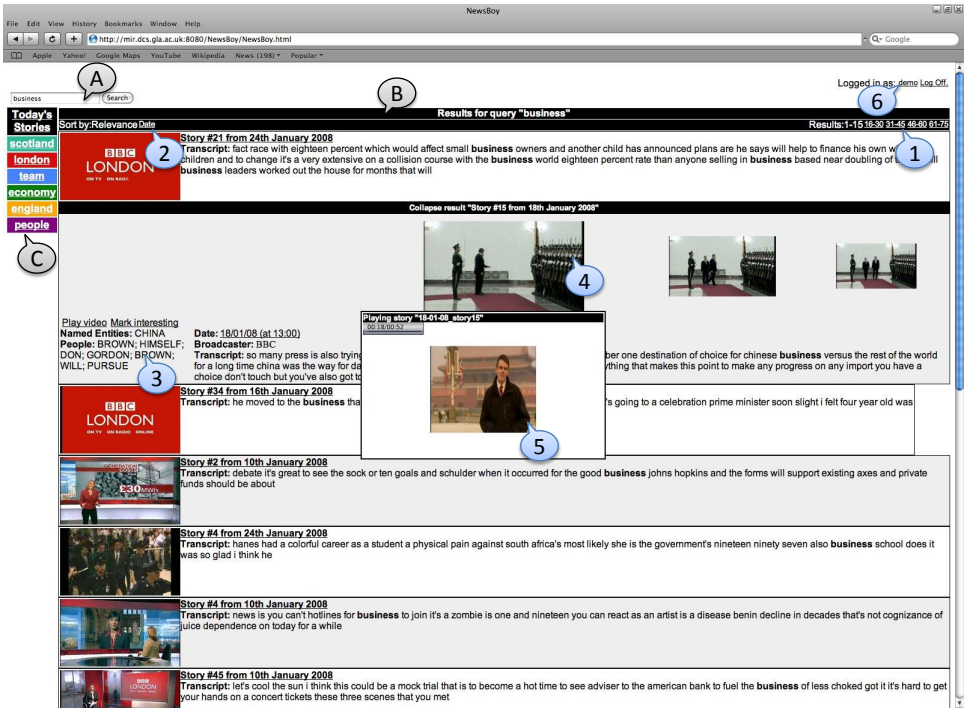

Figure 2: NewsBoy Interface

the size of the keyframe grows larger the closer it is to the focused keyframe. In the expanded display, a user can also select to play a video or to mark it as interesting. Clicking on "play video" starts playing the story video in a new panel (5).

NewsBoy recommends daily news videos based on the user's multi-aspect preferences. These preferences are captured by unobtrusively observing the user's interactions with the NewsBoy interface. By clustering the content of the profile NewsBoy identifies different topics of interest and recommends these topics to the user. The personalisation approach will be introduced in Section 4. The interface presents these topics as labelled clusters on the left hand side of the interface (C). Each cluster represents a group of terms, hence, when a user clicks on the term, a new search is triggered, using the selected terms as a new query. Results are displayed in the result panel.

On the top of the interface, the users can edit their profile by clicking on their username (6). This action will pop up a new frame where the top weighted terms of each cluster are listed, and the user can edit terms or the aligned weighting. Furthermore, the user can manually add new weighted terms.

In this section, we introduced the basic components of a video retrieval system, the frontend and the backend. These components enable the users to explore the indexed data collection. In the next section, we introduce our methodology of enhancing the users' search sessions by adapting the output of the system to their personal interests.

4. PERSONALISATION

The aim of NewsBoy is to deliver daily news videos based on the user's interest in multiple aspects of daily news. This procedure raises some research questions. The main question is how the user's interest can be captured and represented. Furthermore, we are interested how multiple interests can be identified. A common approach is to interpret the user's interactions with the system's interface and to represent this interest in a profile. The process of gathering these interactions will be introduced in Section 4.1. Our approach of incorporating the user's evolving interest will be shown in Section 4.2. In Section 4.3, our approach of identifying multiple interests will be explained.

4.1 Relevance Feedback

O'Sullivan et al. [15] evaluated the use of explicit and implicit relevance feedback to recommend video stories. Their results indicate that user profiles created by exploiting implicit feedback are as valuable as profiles which are created by incorporating explicit feedback only. Hopfgartner and Jose [9] identified various implicit indicators of relevance in video retrieval when comparing the interfaces of state-of-the-art video retrieval tools. The most common features they identified were: clicking on a keyframe to start playing a video, browsing through a result list, using the sliding bar to go through a video, highlighting additional metadata and playing a video for a certain amount of time. However, analysing which of these implicit measures are useful to infer relevance has rarely been done.

NewsBoy tries to capture the users' interests by exploiting the implicit relevance feedback captured from users interacting with the interface introduced in Section 3.2. The interface provides various possibilities to provide implicit relevance feedback. Users interacting with it can:

- Expand the retrieved results by clicking on it.
- Play the video of a retrieved story by clicking on "play

video".

- Play the video for a certain amount of time.
- Browse through the keyframes.
- Highlight additional information by moving the mouse over the keyframes.

Any of these interface features can be seen as a possible indicator of relevance. Which one of these implicit measures are good indicators of relevance is not clear though. While Claypool et al. [7] identified time spend on a web site as being a valid implicit indicator of relevance in the text domain, Kelly and Belkin [11] criticise the time factor as indicator in the video domain. They assume that information-seeking behaviour is not influenced by contextual factors such as topic, task and collection. Their study cast doubt on the straightforward interpretation of dwell time as an indicator of interest or relevance. Hence, we decided to ignore the playing duration as a positive indicator and focus on the remaining indicators only.

Further research has to be done to find the strongest indicators in order to identify optimal an weighting for each interface feature. This is, however, not the focus of this work. Based on the analysis of implicit relevance feedback weight by Hopfgartner and Jose [9], we therefore define a static value for each possible feature:

$$W = \begin{cases} 0.1, & \text{when a user uses the highlighting feature} \\ 0.2, & \text{when a user starts playing a video} \\ 0.3, & \text{when a user browses through the keyframes} \\ 0.5, & \text{when a user expands a result} \end{cases}$$

For capturing the users' interest, NewsBoy extracts the (non-stopword) query terms aligned with the story item a user interacted with, combines them with the feedback weighting and stores the weighted terms in the profile. The following example explains the process: A user retrieved a list of stories and decides to expand the first result. Capturing this action, NewsBoy extracts all terms aligned with this result, combines them with the weighting 0.5 in a vector and submits this vector to the profile. A more in-depth description of this profiling is given in the following section.

4.2 Profile

User profiling is the process of learning the user's interest over a longer period of time. In this section, we introduce our approach of capturing the users' interest and introduce the representation of this interest in the profile. The profile is the fourth component of the NewsBoy system illustrated in Figure 1. Furthermore, we introduce our approaches of representing the user's evolving focus of interest.

4.2.1 Profile Learning and Representation

Several approaches have been studied to capture a user's interest in a profile, the most prominent being the weighted keyword vector approach. In this approach, interests are represented as a vector of weighted terms where each dimension of the vector space represents a term aligned with a weighting. The weighting of the terms will be updated when the system submits a new set of weighted terms to the profile starting a new iteration j. Hence, we represent the interaction I of a user i at iteration j as a vector of weights

$$\vec{I}_{ij} = \{W_{ij1}...W_{ijv}\}$$

where v indexes the word in the whole vocabulary $\mid V \mid$.

We create a weighting W_{ij} by capturing the implicit relevance feedback provided by a user i in the iteration j with the interface introduced in Section 3.2. W has been introduced in detail in Section 4.1. Representative terms from relevant story segments will be extracted and assigned with an indicative weight to each term, which represents its weight in the term space. In our model, we extract non-stopwords v from the stories a user interacted with in the iteration i and assign these terms with the relevance weighting W_{ijv}.

Furthermore, we represent the profile $\vec{P}_i$ of user i as a vector containing the profile weight PW of each term v of the vocabulary:

$$\vec{P}_i = \{PW_{i1}...PW_{iv}\}$$

4.2.2 Ostensive Factor

The simplest approach to create a weighting for each term in the profile is to combine the weighting of the terms over all iterations. This approach is based on the assumption that the user's information interest is static, which is, however, not appropriate in a retrieval context. The users' information need can change within different retrieval sessions.

Campbell and van Rijsbergen [5] propose in their ostensive model that the time factor has to be taken into account, i.e. by modifying the weighting of terms based on the iteration they were added to the user profile. They argue that more recent feedback is a stronger indicator of the user's interest than older feedback. In our profile, the profile weight for each user i is the combination of the weighted terms v over different iterations j: $PW_{iv} = \sum_j a_j W_{ijv}$. We include the ostensive factor, denoted a_j, to introduce different weighting schemes based on the ostensive model. We have experimented with four different functions to calculate the weighting, depending on the nature of aging, the functions will be introduced in the following paragraphs.

4.2.2.1 Constant Weighting.

$$a_j = \frac{1}{j_{max}} \tag{1}$$

The constant weighting function does not influence the ostensive weighting. As Equation 1 illustrates, all terms will be combined equally, ignoring the iteration when a term was added or updated. The constant weighting can be seen as a baseline methodology which does not include any ostensive factor.

4.2.2.2 Exponential Weighting.

$$a_j = \frac{C^j}{\sum_{k=1}^{j_{max}} C^k} \tag{2}$$

The exponential weighting as defined in Equation 2 gives a higher ostensive weighting to terms which has been added or updated in older iterations. It is the most extreme function as the ostensive weighting of earlier iterations decreases distinctly.

4.2.2.3 Linear Weighting.

$$a_j = \frac{Cj}{\sum_{k=1}^{j_{max}} Ck} \tag{3}$$

Equation 3 defines the linear weighting function. The ostensive weighting of earlier iterations decreases linearly. This function linearly reduces the ostensive weighting of earlier iterations.

4.2.2.4 *Inverse Exponential Weighting.*

$$a_j = \frac{1 - C^{-j+1}}{\sum_{k=1}^{j_{max}} 1 - C^{-k+1}} \quad (4)$$

The inverse exponential weighting defined by Equation 4 is the most contained function. Compared to the other introduced functions, the ostensive weighting of early iterations decreases more slowly.

4.3 Capturing Multiple Interests

All components introduced in the previous sections communicate through the NewsBoy Broker, the fifth component of the system illustrated in Figure 1. The task of the broker is to personalise the system by identifying the user's multiple interests in different aspects. Our methodology of identifying these aspects is introduced in the following.

our approach is based on the assumption that news topics consist of a number of *unique* terms which appear in all stories about one topic. News stories about the topic *football* e.g. might consist of unique terms such as "goal", "offside", "match" or "referee". We capture implicit feedback when a user interacts with these stories. The terms of these stories will be extracted and, combined with the implicit weighting, stored in the profile. Hence, as the particular terms are added with the same weighting, they are close neighbours in the profile's vector space.

In this work, we sort the terms in the user's profile according to their profile weighting and identify the terms which have the five biggest distances to the neighbouring terms. We use these identified weighted terms to cluster the remaining profile terms accordingly. Each cluster represents *one* aspect of the user's interest.

The top weighted terms of each cluster are used as a label to visualise the aspect on the left hand side of the NewsBoy interface (marked (C) in Figure 2). In this work, we limited the number of terms to six. Clicking on this label hence triggers a retrieval with the top six weighted terms of this aspect being used as search query. The effect of the different weighting factors on the user's profile will be illustrated in the following section.

4.4 Weighting Effect

In Section 4.2.2, we introduced four different profile weighting approaches that capture the evolving user need by incorporating the ostensive model. In Section 4.3 we introduced our approach of clustering the terms based on this weighting. In this section, we illustrate the effect of the different weighting factors on the user's profile by simulating users interacting with the NewsBoy interface over several days. Simulations are an alternative methodology of evaluating different approaches to user-modelling. In this methodology, we assume that a user is interacting on the system. If such a user is available, he or she will carry out a set of actions to retrieve or look at relevant results. The aim of our simulation will be introduced in the following section.

4.4.1 *Simulated User Interaction*

C	Const. Term	Const. W	Exp. Term	Exp. W	Lin. Term	Lin. W	Inv. Exp. Term	Inv. Exp. W
1	people	1	people	1	news	1	people	1
					flight	0.99		
					fuel	0.99		
					change	0.99		
					connecting	0.99		
					passengers	0.99		
					houston	0.99		
2	christmas	0.74	christmas	0.77	morning	0.20	thousand	0.84
					people	0.19		
3	thousand	0.64	thousand	0.67	recent	0.14	christmas	0.76
					past	0.14		
4	house	0.48	twenty	0.49	sounds	0.11	twenty	0.65
	twenty	0.48	house	0.49	home	0.10		
					company	0.10		
					thousands	0.10		

Table 1: Top four clusters in the simulated user profile for the constant, exponential, linear and inverse exponential ostensive weighting functions.

In order to get an insight into the effect of the four different ostensive weighting functions on the clusters, we simulate a user interacting with stories of each day of our data collection. In a first step, we retrieve all stories on a particular date, starting with the oldest recording available. We then simulate a user interacting with these results by randomly selecting x stories, where $0 \leq x \leq$ (# of stories). In the next step, the simulated user can (a) start playing a video, (b) expand a result and (c) use the highlighting feature. Each of these events has an equal probability of 33%. Furthermore, the simulated user browses up to ten times through the keyframes, each browsed with a probability of 10%. Each simulated action will start the profiling process which has been introduced in Section 4.1. The same simulation is repeated for all days of our data collection. A possible search session i.e. could be: A user expands a result, plays a video and browses through three keyframes.

We are aware that a user does not "randomly" select results and that the probability of using a feature is not always 50%. However, our aim is to evaluate the different ostensive weighting functions, which are not user-dependent. Hence, we decided to base our simulation on a simplified user model.

4.4.2 *Profile Content*

Exploiting the simulated user profile, we clustered the terms based on the different ostensive factors. Table 1 illustrates the four top clusters for the constant, exponential, linear and inverse exponential ostensive weighting functions.

Assuming that the profiles represent the interests of the simulated user, some observations can be stressed when analysing the top terms stored in the profile. While the profiles which are weighted using the *constant*, *exponential* and *inverse exponential* functions show similar clusters, the biggest difference can be spotted in the profile which incorporates the *linear* ostensive function. This profile seems to emphasise latest events, a news story related to air flights. Past events such as christmas do not appear in the top clusters of this profile, however, the term "christmas" has a high ranking in the remaining profiles, indicating that past events are still represented in the clusters.

The example profiles hence confirm our expectation that the introduced ostensive factors will set a different emphasis on added terms, based on the time when the terms were added. However, which of these profiles represents the user's current interest cannot be answered by this study. A meaningful interpretation requires knowledge about the users preferences which can only be achieved by a user study. In the following section, we introduce a subsequent user study we

performed.

5. PRELIMINARY EVALUATION

In order to evaluate which clusters created by the different profiling methodologies can best represent the user's multiple interests, we designed a user-centred evaluation. Nine participants of different nationality volunteered to include NewsBoy as an additional source in their daily news gathering process. For one month, the participants used NewsBoy to browse through the displayed news stories or to discover the data collection.

As we used our own data collection which grows every day and do not provide the users with pre-defined search topcis, an evaluation based on precision and recall, as common in evaluation campaigns such as TRECVID is not possible. Therefore, we aimed to evaluate the models based on the user's satisfaction.

5.1 Participants

The participants were mostly postgraduate students and research assistants with a background in computing science. The group consisted of eight male and one female with an average age of 27.7 years and advanced proficiency with English.

Prior to the experiment, each participant was asked to fill out a questionnaire so that we could measure their experience of dealing with news media and personalisation systems. The most cited topics of interest are sports, politics, science and entertainment. The group follow news by watching television once or twice a month and mainly use the internet as their daily source of information. The BBC and Google News websites were mostly cited as favourite source, followed by websites of national newspapers such as `www.spiegel.de` or `www.elmundo.es`. Furthermore, they use the internet to occasionally watch news videos online. The most common search strategy that the participants mentioned was browsing their favourite websites.

The questionnaire revealed a clear tendency towards news personalisation systems. However, the participants stated that they are sensitive about the type of feedback they have to provide in order to get personalised news. Privacy is considered to be an important issue, hence, the participants would not agree to provide details about the income or the private address. For the group, providing explicit relevance feedback is an unpopular approach, supporting our methodology of relying on implicit relevance feedback to adapt to the user's interests.

Summarising, the participants mostly rely on the internet to follow daily news and are open-minded towards personalisation systems.

5.2 Objective Profile Evaluation

During the study, the users performed an average of 9.8 implicit actions each day which triggered the profiling process introduced in Section 4.3. Hence, the ostensive weighting of the profile terms introduced in Section 4.2.2 is computed over 190 iterations on average which strongly influences the different weighting approaches.

Table 2 shows the average number of terms when clustering the terms of the users' profiles based on their ostensive weighting as introduced in Section 4.3. As can be seen, clustering the terms based on their ostensive weighting results in different clusters, supporting the assumption that the previously introduced ostensive factor influences the users' profile. However, two drawbacks can be seen in the table. First of all, many clusters consist of few words only, indicating that the introduced methodology of identifying multiple interests is not appropriate and needs to be further investigated. Moreover, some clusters consist of a large amount of terms which hardly represent any specific interest of the users.

	Const.		Exp.		Lin.		Inv. Exp.	
C	Mean	Median	Mean	Median	Mean	Median	Mean	Median
1	1	1	1.125	1	5.625	1	1	1
2	1.125	1	2.375	1	9.625	7	1	1
3	2.375	1	7.25	1	16.25	19.5	1.75	2
4	184.375	2	34.125	4	8.375	3.5	2.25	2
5	166.375	7.5	434.5	9.5	24.125	22.5	3.75	2.5

Table 2: Number of terms of the top four clusters C for each ostensive weighting

While the table confirms the effect of the ostensive model on user profiling, a conclusion about the quality of these profiles cannot be drawn. Therefore, we further focused on the users' subjective opinion about the content of the profile. The evaluation will be introduced in the following section.

5.3 Subjective Profile Evaluation

First of all, we were interested to identify which ostensive weighting function best represents the users' interests. Hence, we used the constant weighting factor introduced in Section 4.2.2 to create the user profile. Thus, the weighting of the terms in the profile is not influenced by an ostensive factor. At the end of the experiment, we clustered the user profile based on the exponential, linear and inverse exponential factor, respectively, and asked the participants to judge, which of these clustered profiles represents their information need best.

In the first question, we asked our participants to judge which profile is the most efficient one in clustering the terms in accordance to their semantic meaning. This question was aimed to analyse whether our assumption that news stories consist of a number of unique terms which appear in all stories of the topic can be applied to identify semantically related terms. The participants did not highlight any particular profile, indicating that the different weighting schemes semantically cluster terms in a similar way.

In the next question, we asked them to judge which profile identified best their interest. Here, the participants prefered the profile created using the inverse exponential weighting function, followed by the constant weighting and linear weighting. The exponential weighting received the lowest ranking, indicating that the approach of giving a higher weight to most recent feedback does not cover the user's long term interest.

In a follow up question, we were interested if the order of the clusters in the profiles represent the participants' interest accordingly. Again, the users showed a tendency towards the inverse exponential weighting function, followed by the constant and linear weightings.

Concluding, the questionnaires revealed a slight preference towards the model which privileges most recent feedback.

6. DISCUSSION AND CONCLUSIONS

In this paper, we address two main research challenges in the field of information retrieval. The first problem we

introduce is how to capture and represent a user's evolving information need. As Campbell [4] argues, the users' information need can change within different retrieval sessions and sometimes even within the same session. The user's search direction is directly influenced by the documents retrieved. So far, capturing and representing this dynamic user interest is an unsolved problem. Another question is how the different aspects of a user's interest can be represented. A user can be interested in various aspects, which also might evolve over time.

In order to study these problems, we introduced NewsBoy, a news video retrieval system which delivers news videos based on the user's interest. NewsBoy captures the user's interactions by extracting relevant terms from results a user interacted with. These terms are combined with an explicit relevance weighting and stored in a user profile. A user's interest in multiple aspects is identified by clustering the profile based on this weighting. We introduced four different models that incorporate the ostensive model to capture the evolving interest of the user. For each model we show their effect on user profiling by conducting a simulated user study. In addition to this, we performed a user study to evaluate these models based on the user's satisfaction. The study indicates the user's preferences against the model which privileges most recent feedback.

In conclusion, our results have highlighted that the ostensive model can be incorporated to represent the users' interests in video retrieval. While we present in this paper a preliminary user evaluation, we plan to further analyse the users' feedback, i.e. by exploiting the log files, which should help to investigate in the introduced research questions.

7. ACKNOWLEDGMENTS

This research was supported by the European Commission under the contracts FP6-027026-K-SPACE and FP6-027122-SALERO. It is the view of the authors but not necessarily the view of the community.

8. REFERENCES

[1] R. Arezki, P. Poncelet, G. Dray, and D. W. Pearson. *Adaptive Hypermedia and Adaptive Web-Based Systems*, chapter Web Information Retrieval Based on User Profiles, pages 275–278. Springer Berlin / Heidelberg, 2004.

[2] K. Bharat, T. Kamba, and M. Albers. Personalized, interactive news on the web. *Multimedia Systems*, 6(5):349–358, 1998.

[3] J. S. Boreczky and L. A. Rowe. Comparison of Video Shot Boundary Detection Techniques. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 170–179, 1996.

[4] I. Campbell. Supporting information needs by ostensive definition in an adaptive information space. In *MIRO'95 – Workshops in Computing*. Springer Verlag, 1995.

[5] I. Campbell and C. J. van Rijsbergen. The ostensive model of developing information needs. In *Proc. of CoLIS-96, 2nd Int. Conf. on Conceptions of Library Science*, pages 251–268, 1996.

[6] L. Chen and K. Sycara. WebMate: A personal agent for browsing and searching. In K. P. Sycara and M. Wooldridge, editors, *Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98)*, pages 132–139, New York, 9–13, 1998. ACM Press.

[7] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In *Intelligent User Interfaces*, pages 33–40, 2001.

[8] M. Hancock-Beaulieu and S. Walker. An evaluation of automatic query expansion in an online library catalogue. *J. Doc.*, 48(4):406–421, 1992.

[9] F. Hopfgartner and J. Jose. Evaluating the Implicit Feedback Models for Adaptive Video Retrieval. In *ACM MIR '07*, pages 323–332, 09 2007.

[10] A. Jaimes, M. Christel, S. Gilles, S. Ramesh, and W.-Y. Ma. Multimedia Information Retrieval: What is it, and why isn't anyone using it? In *MIR '05*, pages 3–8, New York, NY, USA, 2005. ACM Press.

[11] D. Kelly and N. J. Belkin. Display time as implicit feedback: understanding task effects. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 377–384, New York, NY, USA, 2004. ACM Press.

[12] D. Kelly and J. Teevan. Implicit Feedback for Inferring User Preference: A Bibliography. *SIGIR Forum*, 32(2), 2003.

[13] H. Lee, A. F. Smeaton, N. E. O'Connor, and B. Smyth. User evaluation of Físchlár-News: An automatic broadcast news delivery system. *ACM Trans. Inf. Syst.*, 24(2):145–189, 2006.

[14] D. M. Nichols. Implicit rating and filtering. In *Proceedings of 5th DELOS Workshop on Filtering and Collaborative Filtering*, pages 31–36. ERCIM, 1998.

[15] D. O'Sullivan, B. Smyth, and D. C. Wilson. Explicit vs Implicit Profiling - A Case-Study in Electronic Programme Guides. In *IJCAI'03 – Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico*, pages 1351–1353, 08 2003.

[16] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC 1994), Gaithersburg, USA*, 1994.

[17] G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. *ACM SIGIR conference on research and development in Information Retrieval*, pages 49–58, 1993.

[18] N. Sebe and Q. Tian. Personalized Multimedia Retrieval: The New Trend? In *MIR '07*, pages 299–306, New York, NY, USA, 2007. ACM.

[19] A. F. Smeaton. The Físchlár Digital Library: Networked Access to a Video Archive of TV News. In *TERENA Networking Conference 2002, Limerick, Ireland, 3-6 June 2002*, 2002.

[20] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press.

[21] A. Spink, H. Greisdorf, and J. Bateman. From highly relevant to not relevant: examining different regions of relevance. *Inf. Process. Manage.*, 34(5):599–621, 1998.