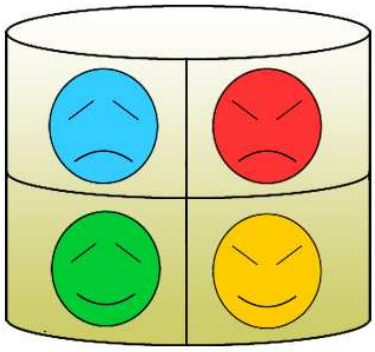
PersDB 2010

4th International Workshop on
Personalized Access, Profile Management, and
Context Awareness in Databases



ELECTRONIC PROCEEDINGS

In conjunction with VLDB 2010

September 13, 2010
Singapore

Foreword

Proliferation of database-driven web sites has brought upon a plethora of information access and dissemination applications. Monitoring and trading stock portfolios, news notification, weather tracking, and even simple search are just a few examples. In addition, recently, new applications have emerged that go beyond information access and dissemination and enhance creativity, information sharing, and collaboration among users providing richer interaction possibilities. Now, users cannot only access content but they can also generate, share and modify content (both theirs and others') freely, compose their applications, enhance their interface, etc. Web-based communities, wikis, social networks, mashups, folksonomies are some of the emerging new applications. In all these (classical and novel) applications, different notions of user information, such as preferences, community memberships and social interactions, and context information, such as a user's social network, location, time, and other features of a user's environment, are of paramount importance in order to improve and personalize user experience. In this context, new challenges emerge for user-centric, context-aware database systems for storing and managing different aspects of user and context information, for data management and computing taking into account personal, social and contextual information about users and for customizability of their behavior.

User-centric, personalized, socially-affected, and context-aware database systems represent a remarkable step towards user-centric applications that allow users to find, generate, share and modify content and services. It is imperative that people studying and working on the different components of a database system clarify their view of contextualization and personalization and describe the ways and degree to which these can be applied to their components of interest. We need a common understanding of the new challenges and we need to design new models, new algorithms, new databases to count for the user-centric requirements of emerging applications. The PersDB 2010 workshop aims at providing a forum for presentation of the latest research results, new technology developments, and new applications in the areas of personalized/socialized access, profile management, and context awareness in database systems.

The program committee consisted of 18 members and was chaired by Tiziana Catarci (Universita di Roma "La Sapienza", Italy) and Yannis Stavrakas (Institute for the Management of Information Systems, Greece). 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 for their interest and participation in PersDB 2010, the reviewers for their time and effort, and the organizers.

Program Chairs

Tiziana Catarci (Universita di Roma "La Sapienza", Italy)
Yannis Stavrakas (Institute for the Management of Information Systems, Greece)

Steering Committee

Tiziana Catarci (Universita di Roma "La Sapienza", Italy)
Yannis Ioannidis (University of Athens, Greece)
Georgia Koutrika (IBM Almaden Research Center, USA)

Workshop Organization

Steering Committee

Tiziana Catarci (Universita di Roma "La Sapienza", Italy)

Yannis Ioannidis (University of Athens, Greece)

Georgia Koutrika (IBM Almaden Research Center, USA)

Program Committee Chairs

Tiziana Catarci (Universita di Roma "La Sapienza", Italy)

Yannis Stavrakas (Institute for the Management of Information Systems, Greece)

Program Committee

Wolf Tilo Balke (University of Hannover, Germany)

Michael Benedikt (Oxford University, UK)

Paolo Ciaccia (University of Bologna, Italy)

Irene Fundulaki (ICS-FORTH, Greece)

Werner Kiessling (University of Augsburg, Germany)

Alexandros Labrinidis (University of Pittsburgh, USA)

Carlo Meghini (CNR, Italy)

Massimo Melucci (University of Padua, Italy)

Mohamed Mokbel (University of Minnesota, USA)

Moira Norrie (ETH-Zentrum, Switzerland)

Panagiotis Papadimitriou (Stanford University, USA)

Christos Papatheodorou (Ionian University, Greece)

Evi Pitoura (University of Ioannina, Greece)

Guillaume Raschia (Polytech' Nantes, France)

Letizia Tanca (Polytecnico di Milano, Italy)

Martin Theobald (Max Plank Institute, Germany)

Xi Zhang (University at Buffalo-SUNY, USA)

Xiaofang Zhou (The University of Queensland, Australia)

PersDB 2010 Website

Anastasios Arvanitis (National Technical University of Athens, Greece)

Table of Contents

A Benchmark for Context Data Management in Mobile Context-Aware Applications*

Nadine Fröhlich Thorsten Möller Steven Rose Heiko Schuldt
Databases and Information Systems Research Group
University of Basel, Switzerland
firstname.lastname@unibas.ch

ABSTRACT

Over the last few years, computational power, storage capacity, and sensing capabilities of mobile devices have significantly improved. As a consequence, they have undergone a rapid development from pure telecommunication devices to small and ubiquitous computing platforms. Most importantly, these devices are able to host context-aware applications, i.e., applications that are automatically adjusted to the current context of their user. This, in turn, requires sensing support on the device, the possibility to store context information, and to efficiently access this context information for the automated adaptation of applications. In this paper, we introduce a benchmark for context management in mobile context-aware applications. We present in detail the design and setup of the benchmark, based on an eHealth use case. The benchmark evaluation considers context queries on Android Nexus One cell phones and compares the performance of different settings including relational and object-oriented databases on the mobile device, and an RDF triple store on a stationary computer. The results show significant differences in the settings that have been evaluated and are thus valuable indicators for database selection and system design for mobile context-aware applications.

1. INTRODUCTION

Over the last few years, mobile devices have undergone a rapid metamorphosis from pure telecommunication devices to small and ubiquitous computing platforms. This is the result of a multitude of technical developments: i.) new types of powerful and energy-efficient processors; ii.) the significant increase in local storage capacity due to inexpensive, low latency flash memory cards; iii.) sophisticated sensing capabilities already embedded into off-the shelf devices (e.g., GPS sensors or acceleration meters). As a result of these developments, mobile devices are more and more in the focus of novel kinds of applications that aim at improving the way users access data and information. A particular emphasis has been put in recent years on *context-aware applications*. These applications aim at automatically adapting the way information is accessed, processed, and/or presented to the current needs of their user, based on their context (e.g., preferences, location, devices at hand). As users are mobile, their context may rapidly change over time. Hence, in contrast to traditional earmarked applications running on stationary devices with users whose context is rather static, mobile applications are characterized by the intrinsic high frequency in which the users' context changes. For the management of sensed context and the exploitation of context for automated adaptation, e.g., by a rule engine, efficient and effective context data management is needed.

Motivation: eHealth Use Case. Consider, for instance, information access in a hospital. On a ward round, the doctor's time should mainly be spent for the interaction with their patients, rather than for searching in the clinical information system. For example, retrieving electronic patient records should be unobtrusive and efficient. When a physician enters a sick room, her mobile phone should automatically display the medical history of the patient(s) in this room. Annotations to the health record added via the device will be synchronized with the underlying clinical information system. In case she wants to share medical images with her patient for which the mobile device's display is too small, the images should be automatically transferred to the patient's bed-mounted multimedia device.

Challenges and Contribution. For the type of dynamic adaptations described before, the current context of a user needs to be sensed and stored for immediate and/or later exploitation. The amount of context data to store can vary significantly, depending on the frequency in which the user's context changes and the frequency in which it is sensed. Although today's mobile devices are much more powerful compared to devices as of some years ago, resources are still limited. Therefore, databases for storing and retrieving context data need to be as efficient as possible. We have designed a benchmark tailored to context data management for mobile applications (aligned to the eHealth use case presented above) in order to identify which data store is best suited for this task. This benchmark has been implemented using three different kinds of open source databases: i.) relational databases locally on the device and on a remote stationary server ii.) an RDF triple store on a remote server, and iii.) an object-oriented database locally on the device and on a

*This work has been partly funded by the Hasler Foundation

PersDB '10, September 13, 2010, Singapore

remote stationary server. We could only run the triple store remotely, with updates and queries submitted from a mobile device, as there is, to the best of our knowledge, currently no stable open source RDF-based implementation directly running on mobile devices. However, this configuration is very relevant, since context information will be subject to reasoning. To factor out communication costs when interacting with a remote server, we also ran the benchmark on the object-database and two relational databases remotely.

For the evaluation we used the context data model designed for the LoCa project [7], implemented for each of the data stores. Furthermore, we implemented a data generator to populate all databases with the same context data. The benchmark queries have been tailored to the different data models and schemas while keeping the original semantics as defined in the benchmark set-up.

The paper is organized as follows: Section 2 reviews related work. The benchmark setup is discussed in Section 3. The implementation of the benchmark is described in Section 4 and Section 5 presents and discusses the evaluation results. Section 6 concludes.

2. RELATED WORK

Existing OLTP and OLAP-style database benchmarks usually address data center-scales rather than mobile environments. The TPC-C and TPC-E Benchmarks [21] give preference to a relational data model, SQL as the query language, and aim at enterprise-scale database systems. Likewise, the Berlin SPARQL Benchmark [3] focusses on RDF triple stores, and the LUBM Benchmark [9] evaluates OWL knowledge base systems, both bound to SPARQL as the query language. The Pole Position [15] benchmark compares relational and object-oriented database engines and object-relational (O/R) mapping implementations, but not in a mobile setting.

In general, benchmarks for mobile environments are rare. There exists an open source benchmark [11] designed for the Android platform which compares the object-oriented database Perst [14] with SQLite. However, the evaluations are too limited to assess the overall performance characteristics of these systems. All queries access only one relation without joins, aggregations, subselects, etc. Therefore, they cannot be directly applied to the kind of schema and queries that have to be considered for context-awareness.

Our benchmark compares the performance of different approaches for storing context data. In [20], context models are compared w.r.t. simplicity and flexibility, while we focus in our work explicitly on performance aspects. [2] identifies requirements on context stores, especially in terms of historical context data, which are considered in our benchmark.

3. BENCHMARK DESIGN

In what follows, we introduce the data model designed for the benchmark and the query mix to be considered (more details on the benchmark specification and evaluation can be found in [8]).

Benchmark Data Model. For the benchmark, we exploit the LoCa data model for context data which is defined in accordance with the most commonly used definition for context (Dey et al. [6]). The LoCa context model is introduced in detail in [7] and summarized in Figure 1 and Table 1.

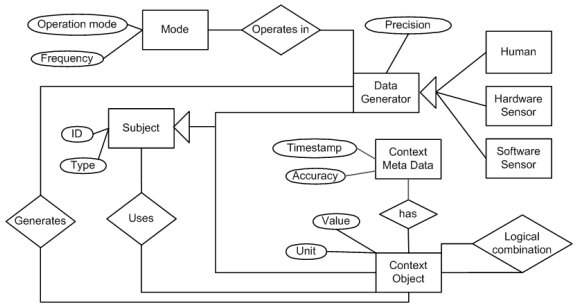


Figure 1: LoCa Context Model

Subject: specifies for whom the context is determined, e.g., the owner of a mobile phone
ContextObject: The context of the subject, e.g., the location of a physician. Context objects have a value, e.g., current GPS coordinates, which have a unit, e.g., degree (for longitude and latitude).
DataGenerator: the sensor that senses context data. Besides hardware (e.g., GPS) and software sensors (e.g., diary) we include manual human input.
Mode: includes metadata on DataGenerators (e.g., in which frequency a generator produces elements of a data stream.
ContextMetaData: metadata on ContextObjects (e.g., timestamp of sensed data, or accuracy of context data).
LogicalCombination: ContextObjects can recursively consist of ContextObjects.

Table 1: Description of the LoCA Context Model

Benchmark Test Load. The objective for the specification of the benchmark test load was to be as realistic as possible (in terms of the different types of queries, their mix, and the volumes of data), without giving preference to a very specific application. For this, we have analyzed typical eHealth environments (information access of physicians in the course of a day, based on statistical information of a medium-sized hospital) and have generalized these findings for the specification of our benchmark. Based on these numbers we estimated the amount of context data to be collected in the course of one year (see Table 2 column 3). On a mobile device, only context data of one user are stored so we scaled the data of column 3 down to one person (see Table 2 column 2). In order to make the benchmark as flexible as possible, we consider three settings with context data of i.) a year, ii.) a quarter, and iii.) a week (in the latter two cases, the numbers from Table 2 are scaled down accordingly).

3.1 Benchmark Queries

The benchmark queries we have defined reflect the way mobile users and context-aware applications interact with a context database. The query mix considers mainly read accesses to context data, as well as insert operations which create new context objects as the user's context evolves.

Q1: Return a subject by a given ID. Such queries are often used, for instance for the identification of a physician.

Q2: Return the last recorded context object of a given type for a subject, e.g., the last blood pressure value.

*Cardinality changes over time.

Entity	data sets per person & year	data sets medium-size hospital/year
DataGenerator	851	15.500
Human	1	5.000
HardwareSensor	833	25.000
SoftwareSensor	17	500
Mode	851	5.000
ContextObject	185.000*	925.000.000*
ContextMetaData	185.000*	925.000.000*
LogicalCombination	185*	925.000*
Subject	1.800	27.000 (patients)

Table 2: Cardinalities of Entities in Benchmark

Query	Percentage	Query	Percentage
Query 1	10 %	Query 7	7 %
Query 2	7 %	Query 8	6 %
Query 3	7 %	Query 9	4 %
Query 4	6 %	Query 10	36 %
Query 5	3 %	Query 11	0.5 %
Query 6	13 %	Query 12	0.5 %

Table 3: Benchmark Query Mix

Q3: Return the context object of a given type for a subject in a given time interval (day, week, month).

Q4: Return the last recorded context object of a given type for a subject, generated by a given generator. It considers details on the sensor, e.g., to find out whether a blood pressure meter has the desired precision.

Q5: Return the available data generators including type and precision that generate context objects of a given type for a subject (e.g., for the selection of a generator for a particular application).

Q6: Return a subject of a given type with the same context object as a given subject, for instance when searching for patients belonging to a sick room or for devices available in the sick room in which the physician is currently situated.

Q7: Return all available types of context objects to a given subject in alphabetical order.

Q8: Return all logical combinations of context data for a given subject. The query result shows which context data are part of other context data.

Q9: Return the number of data generators that generate context data per subject. By this query one is able to control the generators belonging to one subject.

Q10: Insert a context object. This operation reflects all context change of a user, as sensed by the device.

Q11: Update all context objects belonging to one logical combination.

Q12: Delete a context object, e.g., for correcting mistakes of human data generators.

The order of the queries in the mix will be randomly chosen, but their occurrence over a longer interval is determined by the percentages given in Table 3. It should be noted that all queries, except for no. 10 (creation of context object) run sequentially, while the insertion is done automatically in the background, in parallel to the dynamic adaptation.

3.2 Performance Metrics

For the benchmark evaluation, we consider the following metrics, which are aligned with the Berlin benchmark [3].

Metrics for Single Queries

- Average Query Execution Time ($aQET_x$): Average time for executing an individual query of type x ten times with different parameters against the system under test (SUT).
- Minimum/maximum Query Execution Time ($minQET_x$, $maxQET_x$): A lower and upper bound execution time for queries of type x.
- Queries per Second (QpS_x): Average amount of queries of type x that were executed per second. This value is computed from the $aQET_x$ values.

Metrics for Query Mixes

Overall Runtime (oaRT): Overall time it took the test driver to execute a certain amount of queries following the distribution in the mix against the SUT. Thereby, inserts are running in a parallel thread on the device. We decided to process thee runs each with 300 queries each.

4. BENCHMARK SETUP

In this section,we describe in detail the setup of our benchmark evaluation.

4.1 Data Stores

We have selected three different kinds of data stores for the benchmark. All are well tested and stable open source systems that are widely adopted in practice.

- *Relational databases*
  i) H2 (v. 1.2.136) in embedded mode and remotely
  ii) MySQL (v. 5.0.51a) running remotely
  iii) SQLite (version 3.5.9) [19] in embedded mode
- *RDF/OWL triple store* Sesame (v. 2.3.1) [16] with Storage and Inference Layer SwiftOWLIM (3.0 beta 12) [13] remotely (servlet, deployed into apache-tomcat-6.0.24).
- *Object-oriented database* db4o (v. 7.12) [4] in embedded mode and running remotely

Initially, we have chosen only SQLite, Sesame, and db4o for the benchmark. However, Sesame/SwiftOWLIM currently does not run on smart phones and SQLite cannot be used in a client/server (c/s) mode. Therefore, we added H2 to our benchmark as it supports both embedded and c/s mode. This makes results for remote operation comparable among the triple store and the relational databases. Finally, we also evaluated MySQL in c/s mode.

The relational database SQLite demands zero configuration, has a small system footprint and no external dependencies to other libraries which makes it highly appropriate for mobile devices. Furthermore, it is currently used in several applications (e.g., Firefox, Google tools). H2 [10] is an open source Java database also having a small system footprint (about 1 MB) and low memory requirements. It supports disk-based and in-memory databases in embedded and server mode. MySQL [12] is a highly popular and very widely adopted open source database, featuring a rich set of functionality.

As the LoCa approach considers the semantics of services when deciding on dynamic adaptations, we have also chosen a triple store for the benchmark. Out of the available open source RDF triple stores we have evaluated, Sesame has been the most promising tool as it is already widely used [16] and has proven to perform well enough for our purpose [3]. Sesame allows for easy extension via the so-called SAIL-API. As the back-end store implementations shipped with Sesame do not provide the OWL based reasoning we would like to use in our model, we used an alternative third party implementation called SwiftOWLIM, an in-memory triple store which implements rule-based forward-chaining strategy for inferencing and supports a subset of OWL DL.

The object-oriented database db4o has been chosen to avoid the impedance mismatch as the LoCa system is completely implemented in Java.

All tests were performed on Nexus One (N1) mobile phones using Google Android version 2.1 [1] platform. The devices come with 512MB RAM, 512MB ROM, and a Qualcomm QSD8250 CPU with 1GHz. When working in remote mode, instances of Db4o, MySQL, and Sesame/SwiftOWLIM ran on a standard server (Intel Core 2 - 6600 2.4GHz, 4GB RAM, 250GB SATA Seagate ST3250820AS, Ubuntu 9.04 x86_64). Mobile phones were connected to a 802.11b/g 54MBit wireless access point (DLINK DIR615), to which the server was connected via 100MBit Ethernet.

4.2 Data Generator

To supply the different types of databases with equivalent datasets, we implemented a parameterizable data generator that incorporates output modules for every target paradigm. In the first step, the raw data is generated in-memory using an integrated object-oriented data model according to the context data model presented in Figure 1. The actual data values are generated by exchangeable value generators, all relations are picked at random (evenly distributed). In the second step, this data is then transformed into the target data formats by special output modules. An intermediate raw format is stored to be able to create additional outputs later, that are equivalent to the formerly generated ones. Additionally, every output module creates equal sets of query parameter values that are used throughout the benchmark run. This ensures a maximum of comparability between the different platforms.

4.3 Implementation of the Data Model

The schemas for the different data stores are created according to the LoCa context data model. For the relational databases, a standard transformation has been applied. The inheritance is implemented using a horizontal partitioning to avoid joins and decrease execution time.

For the transformation to RDF, an OWL DL ontology has been defined[1]. Entities are directly mapped to concepts (classes), relations among entities are mapped to either one or two object roles (properties) depending on whether both directions can be navigated, and attributes to data roles. Furthermore, we exploited additional modeling expressivity available in OWL: cardinality restriction constructors, disjointness of concepts, and functional, transitive, and irreflexive properties whenever appropriate.

For db4o, we transformed the context model to a Java class structure by mapping entities to classes and by using collections for the relationships. To work in an object oriented manner, we applied the composite pattern to implement the inheritance hierarchy.

[1]Available via `http://on.cs.unibas.ch/owl/1.0/Context.owl`

4.4 Implementation of Benchmark Queries

All benchmark queries have been formulated in the query languages supported by the respective data stores. For relational databases we used SQL, for db4o SODA, and for Sesame SPARQL.

In case of H2 and MySQL, we used suitable JDBC drivers to access the databases. SQLite comes as built-in embedded database with its own API that offers different ways to access the database, raw query (nearly plain SQL), and a structured interface for users with little SQL knowledge. For the benchmark, we used the raw query interface.

SPARQL has evolved as the de facto standard for querying RDF graph data. At the moment, there exist two major versions of SPARQL that are specified by the W3C. SPARQL 1.0 [17] has matured to a W3C recommendation and contains basic graph query constructs. At the time of writing SPARQL 1.1 [18] is still a working draft. It adds more advanced features like aggregations and data manipulation to the language. As our chosen triple-store only supports SPARQL 1.0, queries that change the dataset were difficult to handle. While query 12 can be easily implemented using a Sesame specific type of request ('transaction'), query 11 would have required to programmatically perform needed updates on the datasets. Thus, we chose to drop this query in the triple-store implementation of our benchmark. Also aggregating queries had to be implemented partly on the client side. To access the RESTful SPARQL interface that SESAME provides, we used the HttpClient implementation integrated into the Android SDK. All result data was encoded as JSON and parsed using androids own parser implementation.

Db4o supports at its APIs query by example (QbE), Native Queries (NQ) and SODA. We decided to use SODA because it is, according to the db4o documentation [5], up to two times faster than optimized NQ and five times faster than unoptimized NQ — the reason for this is that SODA is the underlying internal query API all the other query APIs are mapped to.

5. BENCHMARK RESULTS

This section summarizes the evaluation results for our benchmark.

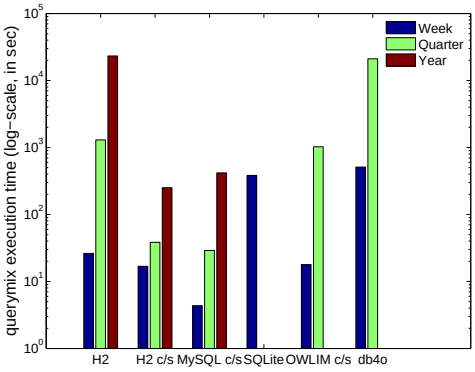


Figure 2: Overview of Query Mixes

System	Week	Quarter	Year
H2 emb	26 (0.9)	1300 (23.8)	23251 (267.3)
H2 c/s	17 (0.9)	38 (23.8)	251 (267.3)
MySQL c/s	4 (0.8)	29 (18.8)	418 (186.0)
SQLite emb	386 (0.8)	– (17.5)	– (172.9)
SQLite emb (r)	384 (0.8)	– (17.5)	– (172.9)
OWLIM c/s	18 (1.6)	1025 (38.1)	– (384.1)
db4o emb	511 (1.5)	21100 (22.3)	– (126.5)
db4o c/s	– (1.5)	– (22.3)	– (126.5)

Table 4: Execution Times of Query Mixes in Seconds (in parentheses: Database Sizes in MB)

Query Mixes. Figure 2 and Table 4 show the results of the query mix evaluation in the different settings. The execution times of the relational databases vary considerably but except for SQLite, all query mixes could be processed. Only for H2 emb, we had to increase the JVM heap size to 80 MB which is however a rather unrealistic setting for the device. SQLite was comparably slow. H2 c/s and MySQL c/s manage to execute the query mix for one year in nearly the same time as SQLite needs for the query mix for one week (SQLite week: 386s, H2 year: 251s, MySQL year: 417s). This is most probably due to the lack of a sophisticated query optimizer for SQLite. Furthermore, H2 c/s proves to better perform on the largest data set (year), but MySQL c/s performs much better on the smaller data sets (week/quarter). As expected, the c/s architectures perform better than the embedded databases on our resource limited mobile devices. For SQLite, we have compared a setting with referential integrity with a setting without. The benchmarks results show only marginal differences. OWLIM c/s shows only an average performance, only slightly better than H2 emb – however, it is not able to process a query mix on the large data set (year). For db4o we could not manage to process a complete query mix. With db4o c/s, query 4 could not be executed although the query ran fine in the db4o embedded version – this seems to be the result of a mashaling/unmarshaling problem in db4o. The db4o emb version is slow, and needs a lot of space and JVM heap.

Single Queries. To find out why the query mixes showed significant performance variations on different systems, we have also analyzed the average execution time of frequently occurring queries. Query 6 has a fraction of 13% in the query mix. For this query, the triple store performs much better than most relational databases and even db4o c/s is for the large dataset (year) in the range of MySQL (see Figure3).

Query 1 has also a major influence on the query mix (10%). SQLite performed best for this query while db4o emb performed comparably bad. db4o c/s performed better but could not reach the times of the relational databases; its performance is comparable to OWLIM (see Figure 4).

Query 9 is one of the most controversial queries in the benchmark since some systems have not been able to properly execute it in reasonable time (see Figure 5). This query runs fine on H2 c/s and MySQL but lasted extremely long on SQLite. One reason for this is might be the less elaborated optimizer of SQLite. H2 could perform this query but we needed to increase the JVM heap to 64 MB. OWLIM performs fair on small data sets but gets very slow with increased data sets.

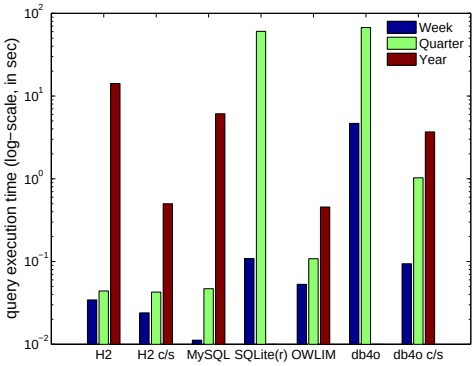


Figure 3: Single Query Evaluation for Query 6

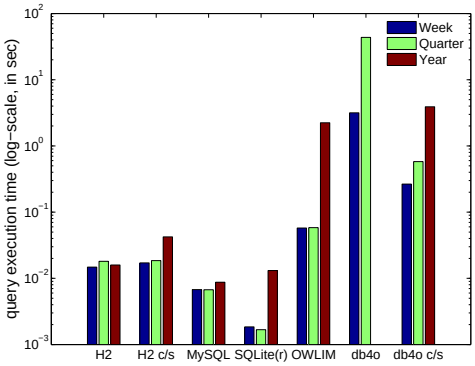


Figure 4: Single Query Evaluation for Query 1

Summary Results. The analysis of the performance for individual queries shows that the chosen systems have different strengths. A system that is very slow for one query can be fast for another query. For deciding what system is the best we have to care about the overall performance shown in the query mixes where the fraction of queries is considered. In the mixes, read-only queries (without Query 10) are running in parallel to inserts (Query 10). The execution of inserts usually lasts a few milliseconds (2 to 35 ms), except for OWLIM where the execution of inserts lasts more than 10000 milliseconds. The performance in c/s mode was mostly considerably better than in embedded mode. H2 needed for the query mix with the context data accumulated in one year 23251s while H2 c/s needed 251s. For db4o we could not run a complete query mix. But when comparing the single queries it is obvious that db4o c/s is mostly faster

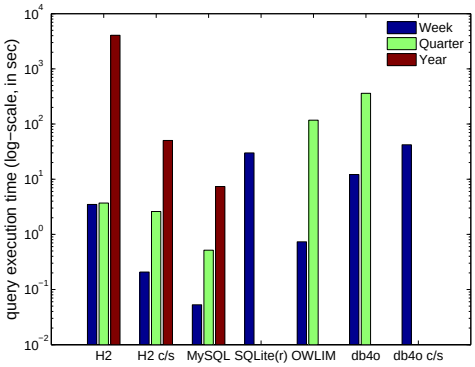


Figure 5: Single Query Evaluation for Query 9

than db4o embedded. Query 9 seems to be an outlier as it performs better when executed in embedded mode than in c/s mode (see Figure 5). Although additional communication cost incur in c/s mode, powerful server-side hardware can partly or completely compensate this. Only for db4o, communication costs are in some cases higher than the benefit of more powerful servers. Finally, the benchmark shows that it is not feasible to store context data for a longer period in an embedded database on a mobile device.

Lessons Learned. *SQLite* is small and easy to use, but its optimizer is not as developed as that of more mature database systems. We had to manually optimize some of the queries to improve the query performance.

H2 supports referential integrity by default and works well in embedded and client/server setting. Most notable, H2 caused least configuration problems.

MySQL, as the data store with the richest set of features, is also the most demanding system regarding configuration for remote access.

OWLIM, Sesame's built-in data store, can be easily created and maintained using the web based workbench. However, manual configuration was needed to create repositories working with an OWLIM store after the binaries were integrated into the Servlet distribution. As we chose the RESTful HTTP interface to access the stores, some conveniences of common database access layers had to be implemented on top of the standard http client, e.g., to circumvent memory restrictions when large result sets were received. This was the case for Query 8, where a lacking aggregation feature of SPARQL 1.0 had to be implemented on the client side.

db4o was comparatively slow, especially for Query 9 and it demanded much memory and JVM heap space (up to 64M/80M). For speeding up the tests, we used the SODA query language that is much faster than the recommended query interface (NQ). Actually, the performance of db4o also strongly depends on the design of the data model as inheritance and collections with a lot of associated objects slow down the query execution dramatically. We optimized our model, but in order to reach an appropriate speed we would have to completely redesign our model, especially by abandoning the object-oriented design (inheritance). In c/s mode, one query (no. 3), did not run at all, although we did not experience any problems in embedded mode. Although db4o performs well on a standard PC in c/s mode, we were faced with major problems regarding performance and stability in embedded mode on the mobile device. These problems are well taken by the db4o developers as they are currently aiming at decreasing the needed stack size to better support Android-based platforms.

6. CONCLUSIONS AND FUTURE WORK

Managing context data on mobile devices is an essential prerequisite for supporting dynamic, context-aware adaptations of applications. In the paper, we have presented a benchmark designed for the management of context data and we have reported in detail on the benchmark evaluation which considers relational and object-oriented databases and a triple store in embedded and/or client/server mode. The set-up considers different context data sets accumulated by mobile users on their devices in a realistic setting in the course of a week, a month, and a year. Relational data stores in c/s mode performed best in the benchmark. For smaller data sets, the performance of embedded relational databases and the triple store (c/s) is sufficient.

In our future work, we will focus on the LoCa rule engine to provide support for the automatic adaptation of applications (workflows) and user interfaces, based on the local context store evaluated in our benchmark.

7. REFERENCES

[1] Android Open Source Project. http://source.android.com/.

[2] M. Baldauf and S. Dustdar. A survey on context-aware systems. *Int. Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, June 2007.

[3] Berlin SPARQL Benchmark (BSBM), 2009. http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark.

[4] db4o. http://www.db4o.com/.

[5] db4o Reference. http://developer.db4o.com/Documentation/Reference/db4o-7.12/java/reference/.

[6] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, February 2001.

[7] N. Fröhlich, A. Meier, T. Möller, M. Savini, H. Schuldt, and J. Vogt. LoCa – Towards a Context-aware Infrastructure for eHealth Applications. In *Proc. of the 15th Int'l Conference on Distributed Multimedia Systems (DMS'09)*, 2009.

[8] N. Fröhlich, T. Möller, S. Rose, and H. Schuldt. A Benchmark for Context Data Management in Mobile Applications. Technical Report CS-2010-002, Univ. Basel, 2010. http://informatik.unibas.ch/research/publications_tec_report.html.

[9] Y. Guo, Z. Pan, and J. Heflin. LUBM: A Benchmark for OWL Knowledge Base Systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):158–182, October 2005.

[10] H2 Database Engine. http://www.h2database.com/html/main.html.

[11] McObject Benchmarks Embedded Databases on Android Smartphone. http://www.mcobject.com/march9/2009.

[12] MySql. http://mysql.com/.

[13] OWLIM. http://www.ontotext.com/owlim/.

[14] Perst – An Open Source, Object-oriented Embedded Database. http://www.mcobject.com/perst/.

[15] PolePosition – Open Source Database Benchmark. http://polepos.sourceforge.net/.

[16] Sesame – Open Source Framework for Storage, Inferencing and Querying of RDF Data. http://www.openrdf.org/.

[17] SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/.

[18] SPARQL 1.1 Query Language. http://www.w3.org/TR/2010/WD-sparql11-query-20100601/.

[19] SQLite. http://www.sqlite.org/.

[20] T. Strang and C. Linnhoff-Popien. A context modeling survey. In *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004, Nottingham/England*, 2004.

[21] TPC - Transaction Processing Performance Council. http://www.tpc.org/.

Combining Preference Relations: Completeness and Consistency

Nicolas Spyratos
Université Paris-Sud
Laboratoire de Recherche en Informatique
Orsay Cedex, France
spyratos@lri.fr

Carlo Meghini
Consiglio Nazionale delle Ricerche
Istituto della Scienza e delle Tecnologie della Informazione
Pisa, Italy
meghini@isti.cnr.it

ABSTRACT

We introduce two criteria for judging "goodness" of the result when combining preference relations in information systems: completeness and consistency. Completeness requires that the result must be the union of all preference relations, while consistency requires that the result must be an acyclic relation. In other words, completeness requires that the result contain all pairs appearing in the preference relations, and only those pairs; while consistency requires that for every pair (x, y) in the result, it must be able to decide which of x and y is preferred to the other. Obviously, when combining preference relations, there is little hope for the result to satisfy both requirements. In this paper, we classify the various methods for combining preference relations, based on the degree to which the result satisfies completeness and consistency. Our results hold independently of the nature of preference relations (quantitative or qualitative); and also independently of the preference elicitation method (*i.e.* whether the preference relations are obtained by the system using query-log analysis or whether the user states preferences explicitly). Moreover, we assume no constraints whatsoever on the preference relations themselves (such as transitivity, strict ordering and the like).

1. INTRODUCTION

The problem considered in this paper is how to combine preferences to arrive at a consensus in the context of databases and information systems. Combining preferences to arrive at a consensus is a problem known in the literature as Social Choice Theory [20]. Variants of this problem, such as voting schemes, have been studied since the 18th century (by Condorcet and Borda). More recently, preferences have been used since the 50s in decision making for ranking alternative choices (see [18] for an extensive survey). In databases and information systems, however, the use of preferences started only in the late 90s and it is mainly concerned with the ranking of query answers [13, 7, 19, 1, 6, 4, 5, 12, 15, 14]. Indeed, with the explosion of the amount of information available today (*e.g.*, on the web) query results may be very large, and ranking these results according to users' preferences is of great help to users.

In the area of information systems, user preferences are classified from different points of view, as follows.

In terms of their nature, preferences can be:

- *Quantitative* (or absolute), expressed by a number on a scale (thus capturing intensity of desire). For example, "I like BMW cars 80%" and "I like VW cars 70%". Quantitative preferences are difficult to express by the casual user, but easy to compute by a machine from query logs.
- *Qualitative* (or relative), expressed by comparison. For example, "I like BMW more than VW". Qualitative preferences express no intensity of desire; they are easy to express by the casual user and also easy to infer by a machine (from quantitative preferences).

In terms of their duration in time, preferences can be:

- *Long-term* preferences; these may be either discovered by the system (unobtrusively, from query logs) or declared explicitly by the user. In both cases long-term preferences are stored in the so-called user profile.
- *Short-term* preferences; these are expressed explicitly by the user, on-line, together with a query (in which case one usually talks about *preference queries*).

We note that the nature and the duration in time are orthogonal characteristics of preferences.

In this paper, we assume a set of objects O and we model a preference over O as a pair (o, o') meaning that o is preferred to o'; moreover, we refer to a set of preferences over O as a *preference relation.*

We stress the fact that we model preference relations as binary relations over O *without* any particular constraint such as transitivity, strict ordering and the like. The reason for doing so is that there is no general consensus in the literature as to what properties a preference relation should satisfy. For instance, although transitivity is generally considered as a desirable property, in several situations preference relations are assumed to be non-transitive [9, 10, 11, 22].

Our approach considers only positive preference statements of the kind "x is preferred to y"; in other words, we do not take into consideration indifference relations [18].

VLDB '07, September 23-28, 2007, Vienna, Austria.

When O represents a set of alternative choices, one or more experts may be asked to express their opinion directly on the alternatives, thus leading to a set of preference relations over O. In other cases, the experts may be asked to express their opinion on one or more features of the alternatives, each one inducing a preference relation over O. In both cases, we end up with a set of preference relations over O, and the problem is how to combine them into a single preference relation incorporating as best as possible the opinions of all experts.

In the area of databases and information systems, there are two general methods for combining a set of preference relations: the Prioritized method and the Pareto method, in their restricted and unrestricted versions. Several studies in the literature use these methods, for example for defining preference queries and sky-line queries [3]. However, to our knowledge, no previous work has addressed the problem of how well the information content of the individual preferences is incorporated in the combined relation by these methods.

In this paper, we propose to evaluate Prioritized and Pareto, and their variants, by introducing two criteria: completeness and consistency; and we classify these methods based on the degree to which the combined relations satisfy the two criteria.

Our results hold independently of the nature of preference relations (*i.e.* whether they are qualitative or whether they have been inferred from quantitative rankings); and also independently of the preference elicitation method (*i.e.* whether the preference relations have been obtained by the system, using query-log analysis, or whether they have been stated by the user states, explicitly).

Section 2 gives the formal statement of the problem, as it appears in the area of information systems, also providing some rationale; section 3 introduces the classical methods for combining preference relations; section 4 analyzes how these methods behave with respect to the measures introduced earlier, namely consistency and completeness. Finally, Section 5 summarizes the results and draws some conclusions. Proofs are omitted for reasons of space.

2. STATEMENT OF THE PROBLEM AND RATIONALE

We say that a binary relation P is *complete* with respect to $n > 1$ given binary relations $P_1, \ldots, P_n$ iff P is the union of $P_1, \ldots, P_n$. Moreover, we say that P is *consistent* iff it is acyclic. We note that both completeness and consistency [8] can be tested efficiently.

Given a set of preference relations $P_1, \ldots, P_n$, the problem that we consider is how to find a *combined* preference relation P that satisfies the following requirements:

1. *Completeness.* This property requires that the result should contain all pairs appearing in the preference relations, and only those pairs (*i.e.*, no preference expressed by the user is lost, and no extraneous preference is introduced in the result).

2. *Consistency.* This property requires that the result must be an acyclic relation; that is, for every pair (x, y) appearing in the result, it must be able to decide which of x and y is preferred to the other.

Clearly, when the result is complete, all expressed preferences are taken into account, but there may be contradictions among the individual preference relations, generating cycles in their union. On the other hand, consistency expresses absence of contradictions. So the presence of both completeness and consistency characterizes the optimal situation, where all preferences are taken into account and there is no contradiction.

We note that the presence of contradictions is natural, as they are the result of putting together preferences which either come from different users, independently, or come from the same user but address different aspects of the objects. As we mentioned earlier, such non-contradiction is expressed by the absence of cycles. Indeed, acyclicity allows the ranking of objects, as follows [21, 17]. Let P be an acyclic binary relation (viewed as a digraph), and define the rank of an object o as follows:

- if o is a root of P then $rank(o) = 0$ (a root is a node with no incoming edge);

- else $rank(o)$ is the length of a maximal path among all paths from a root of P to o.

The intuition behind this definition of rank is that the farther an object o is from the best objects (represented by the roots) the less preferred it is. We note that the computational complexity of computing ranks following this definition is linear in the size of the preference graph (understood as usual as the number of nodes plus the number of arcs). In reality, the size of the preference relations is quite small as users often express just a few preferences. Clearly, the definition of rank is sound only if P has at least one root, and this is guaranteed only when P is acyclic.

Now, let us denote by B_i the set of objects with rank i, and let m be the maximal path length among all paths starting from a root. Then it is easy to prove the following proposition:

PROPOSITION 1. *The sequence $B_0, B_1, \ldots, B_m$ defined above has the following properties:*

- *$B_0, B_1, \ldots, B_m$ is a partition of the set of objects appearing in P;*

- *for each $i = 0, 1, 2, \ldots, m$, there is no arc of P connecting two objects of B_i;*

- *for each $i = 1, 2, \ldots, m$ and each object $t \in B_i$ there is an object $s \in B_{i-1}$ such that there exists an arc from s to t in P.*

In general, when combining preference relations, there is little hope for the result to satisfy both completeness and consistency. Since ranking is important in information systems and acyclicity is a sufficient condition in order to do ranking, a reasonable approach to follow when combining preference relations is to satisfy acyclicity while *minimizing* the loss of completeness. In order to achieve this goal, we must select as the result P of combining the given preference relations a largest acyclic subset of the union. In other words, we must select P in such a way that there is no proper superset of P which is an acyclic subset of the union.

If we view preference relations as digraphs, finding such a P is equivalent to solving the maximum acyclic sub-graph problem, which is stated as follows: Given a digraph $G =$

(V, E), find a maximum cardinality subset $E' \subseteq E$ such that (V, E') is acyclic. This problem is known to be NP-hard [16].

Now, since there have been several proposals in the literature for combining preference relations, one wonders how these proposals relate to the maximum acyclic sub-graph problem. This is the question that we undertake in the rest of this paper.

More specifically, we show that the classical methods for combining preference relations provide a clever trade-off between maximality of the result and efficiency of computation. As these methods were introduced long before the theory of complexity was formulated, this is a rather surprising result that provides an *a posteriori* justification for the introduction of these methods.

3. CLASSICAL METHODS FOR COMBINING PREFERENCE RELATIONS

There are several well-known methods for combining a set of preference relations $P_1, \ldots, P_n$, into a single preference relation, most notably, the so-called Prioritized and Pareto (and their variants). In order to formally define these methods, we use the following terminology from [2]. Let P be a preference relation:

- we use interchangeably the notations $(x, y) \in P$ and xPy; if $(x, y) \notin P$, we write $x\overline{P}y$;
- x and y are said to be *equivalent* with respect to P, written $xP^{=}y$, iff both xPy and yPx hold;
- if $x\overline{P}y$ and $y\overline{P}x$, we say that x and y are *incomparable* with respect to P, written $xP^{\#}y$;
- finally, if xPy and $y\overline{P}x$, x is said to be *strictly preferred* to y with respect to P, written $xP^{<}y$.

We first recall the definition of $P_\cup$, the union of the given preference relations $P_1, \ldots, P_n$:

$$xP_\cup y \iff \exists i.(xP_i y)$$

A basic difference between Prioritized and Pareto is that the former, apart from the given preference relations $P_1, \ldots, P_n$, requires some additional information in order to be applied, whereas the latter requires no additional information. Indeed, in the case of Prioritized, one assumes that the preference relations $P_1, \ldots, P_n$, are ordered by a priority relation $\prec$, which is a strict partial order. We recall that a strict partial order is a binary relation which is irreflexive and transitive, and consequently asymmetric (asymmetric means that if $a < b$ holds then $b < a$ does not hold).

Prioritized and Pareto each come into two variants:

1. *Restricted Prioritized* (RPR, for short).

   This is the "classical" Prioritized rule, also used in [2]. Given a set of preference relations $P_1, \ldots, P_n$ strictly ordered by $\prec$, the RPR rule defines a binary relation $Pr_r(P_1, \ldots, P_n, \prec)$, or simply Pr_r when there is no ambiguity, as follows:

   $$xPr_r y \iff \forall i.(xP_i y \vee \exists j.(j \prec i \wedge xP_j^{<} y))$$

   Notice that x and y are incomparable in the combined relation (*i.e.*, $xPr_r^{\#}y$) if and only if x and y are incomparable on the preference relation with the highest priority for which they are not equivalent.

   We note that the well-known lexicographic ordering is a special case of Restricted Prioritized. Indeed, the lexicographic ordering is a Prioritized ordering, in which:

   - O stands for a set of words of finite length over some finite alphabet A, where A is totally ordered by some strict total order $<_A$ (*i.e.*, given any two distinct letters a and a', either $a <_A a'$ or $a' <_A a$ but not both); the minimum element of $<_A$ is the special character *blank;*
   - we have as many preference relations P_i as the number of characters in the longest word in O;
   - the i-th preference relation captures preference based on the i-th letter of words (starting from the left), according to $<_A$; so, $carlo P_1 nicolas$ while $nicolas P_3 carlo$;
   - the priority relation over the P_i's is the natural total order $<_N$ over the indices.

   Under these assumptions, given two words $w = a_1 a_2 \ldots a_m$ and $w' = a'_1 a'_2 \ldots a'_n$, the RPR rule becomes:

   $$wPr_r w' \iff \forall i.(a_i <_A a'_i \vee \exists j.(j <_N i \wedge a_j <_A^{<} a'_j))$$

   Now, since $<_A$ is a strict total order, $<_A = <_A^{<}$, therefore the rule becomes:

   $$wPr_r w' \iff \forall i.(a_i <_A a'_i \vee \exists j.(j < i \wedge a_j <_A a'_j))$$

   The last rule is clearly the rule used to order words in a dictionary.

2. *Unrestricted Prioritized* (UPR, for short).

   This method extends RPR by allowing the combined preference to hold even in presence of incomparability in some preference relation, as long as there exists comparability in some other relation. The UPR rule defines a binary relation Pr_u as follows:

   $$xPr_u y \iff \forall i.(xP_i y \vee (xP_i^{\#} y \wedge \exists k.(xP_k y)) \vee \exists j.(j \prec i \wedge xP_j^{<} y))$$

   Notice that x and y are incomparable if and only if x and y are incomparable in *all* given preference relations.

3. *Restricted Pareto* (RPA, for short).

   In this method, x is preferred to y in the combined relation if and only if for every i, x is preferred to y, and for at least one P_i x is strictly preferred to y. Formally, the RPA rule defines a binary relation Pa_r as follows:

   $$xPa_r y \iff \forall i.(xP_i y) \wedge \exists j.(xP_j^{<} y)$$

4. *Unrestricted Pareto* (UPA, for short).

   In this method, x is preferred to y if and only if for at least one P_i x is strictly preferred to y, and for no j y is strictly preferred to x (in all other preferences x and y can be comparable in the same direction, incomparable or equivalent). The UPA rule defines a binary relation Pa_u as follows:

   $$xPa_u y \iff \forall i.(y\overline{P_i^{<}}x) \wedge \exists j.(xP_j^{<} y)$$

Based on the definitions given so far we can state the following proposition.

PROPOSITION 2. *Let $P_1, \ldots, P_n$, be n preference relations and let $\prec$ be any strict ordering on them. Then, the following hold:*

1. $Pa_r \subseteq Pr_r \subseteq Pr_u \subseteq P_\cup$
2. $Pa_r \subseteq Pa_u \subseteq Pr_u \subseteq P_\cup$
3. *Pa_u and Pr_r are incomparable with respect to set-containment.*

4. COMPLETENESS AND CONSISTENCY OF CLASSICAL METHODS

In order to characterize the classical methods with respect to completeness and consistency, we consider three cases as follows:

Case 1 the union $P_\cup$ is acyclic.

In this case, consistency is guaranteed and we show that completeness holds only for the unrestricted methods.

Case 2 the union $P_\cup$ is cyclic but each individual preference relation is acyclic.

In this case, we show that the restricted methods lead to an acyclic result, but they are incomplete in the sense that they may not produce a maximum acyclic sub-graph; on the other hand, the unrestricted methods may produce a cyclic result.

Case 3 one or more individual preference relations are cyclic.

In this case, all methods may produce a cyclic result.

4.1 Acyclic union

It follows from proposition 2 that whenever the union of the preference relations is acyclic, so are all the variants of Prioritized and Pareto defined above. In this case, then, consistency is satisfied. It remains to be seen whether completeness is satisfied as well.

The next lemma states a consequence of the acyclicity of the union that will be used in what follows.

LEMMA 1. *Let $P_1, \ldots, P_n$, be preference relations whose union $P_\cup$ is acyclic. If $xP_\cup y$ then for some i we have that $xP_i^< y$, and for all $k \neq i$ we have that either $xP_k y$ or $xP_k^\# y$.*

We can now prove the following.

PROPOSITION 3. *Let $P_1, \ldots, P_n$ be preference relations whose union $P_\cup$ is acyclic. Then,*

1. $P_\cup = Pa_u$
2. *$P_\cup = Pr_u$ for any strict order $\prec$ of the preference relations.*

Notice that in proving that $Pr_u \subseteq P_\cup$, the acyclicity of the union is not required.

Concerning Pr_r and Pa_r, we have that in some cases $P_\cup$ is acyclic and that $Pr_r \subset P_\cup$ and $Pa_r \subset P_\cup$. In proof, let us consider the example shown in Figure 1; using RPR to combine P_1 and P_2 leads to the incomparability of 3 and 4 (if $P_1 \prec P_2$) or the incomparability of 1 and 2 (if $P_2 \prec P_1$),

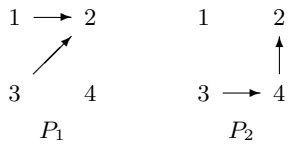


Figure 1: Preference relations

while both these pairs are in $P_\cup = P_1 \cup P_2$. Analogously, it can be seen that $(3,4) \in Pa_r^\#$ and $(1,2) \in Pa_r^\#$.

We can then conclude that when the union of the given preference relations is acyclic, both RPR and RPA are incomplete, while UPR and UPA are "optimal", *i.e.* both complete and consistent.

4.2 Cyclic union with acyclic preference relations

We now consider the case in which the union of the preference relations is cyclic while each individual preference relation is acyclic. In this case, we are interested to know (a) whether Prioritized and Pareto produce an acyclic combined preference relation, and, if yes, (b) how close to the union $P_\cup$ they are.

In order to answer these questions, we first derive necessary and sufficient conditions for the acyclicity of the result in each of the classical methods. We then apply these conditions to answer the above questions.

We recall that a cycle in a binary relation P is a sequence of objects $C = (o_0, o_1, \ldots, o_k)$ with $k \geq 2$, such that

- $o_0 = o_k$,
- $o_{i-1} P o_i$ for each $i = 1, \ldots, k$, and
- there is no repetition in $o_0, o_1, \ldots, o_{k-1}$.

PROPOSITION 4. *Let $P_1, \ldots, P_n$, be n preference relations and let $\prec$ be any strict ordering on them. Then, the Restricted Prioritized Pr_r is acyclic iff for each cycle $C = (o_0, \ldots, o_k)$ in $P_\cup$ there exists an arc (o_{u-1}, o_u) such that for some preference relation P_i, $o_{u-1} \overline{P_i} o_u$ and for each j such that $o_{u-1} P_j o_u$, either $o_u P_j o_{u-1}$ or $i \prec j$.*

The last Proposition states the condition under which an undesired preference (o, o') (*i.e.*, any one of the preferences found in a cycle in $P_\cup$) does not end up in Pr_r. The condition is derived by combining two facts: $oP_i o'$ and $o\overline{Pr_r}o'$ (the latter in turn obtained by negating the RPR rule). By a similar technique, we obtain the analogous conditions for the other considered variants of Prioritized and Pareto. The proofs of the corresponding Propositions are omitted, as they are very similar to that of the previous Proposition.

PROPOSITION 5. *Let $P_1, \ldots, P_n$, be n preference relations and let $\prec$ be any strict ordering on them. Then, the Unrestricted Prioritized Pr_u is acyclic iff for each cycle $C = (o_0, \ldots, o_k)$ in $P_\cup$ there exists one arc (o_{u-1}, o_u) such that for some preference relation P_i, $o_u P_i^< o_{u-1}$ and for each j such that $o_{u-1} P_j o_u$, either $o_u P_j o_{u-1}$ or $i \prec j$.*

PROPOSITION 6. *Let $P_1, \ldots, P_n$, be n preference relations. Then, the Restricted Pareto preference relation Pa_r is acyclic iff for each cycle $C = (o_0, \ldots, o_k)$ in $P_\cup$ there exists one arc (o_{u-1}, o_u) such that either $o_{u-1} \overline{P_i} o_u$ for some preference relation P_i, or $o_u P_j o_{u-1}$ for all preference relations P_j.*

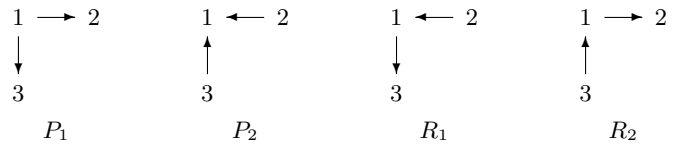


Figure 2: Prioritized combinations

PROPOSITION 7. *Let $P_1, \ldots, P_n$, be n preference relations. Then, the Unrestricted Pareto Pa_u is acyclic iff for each cycle $C = (o_0, \ldots, o_k)$ in $P_\cup$ there exists one arc (o_{u-1}, o_u) such that $o_u P_i^< o_{u-1}$ for some preference relation P_i or $o_{u-1} P_j o_u$ implies $o_u P_j o_{u-1}$) for all j.*

Having established under which condition each of the considered methods produces a cyclic relation, we now examine whether this condition can ever occur.

PROPOSITION 8. *The Restricted Prioritized Pr_r of n acyclic preference relations $P_1, \ldots, P_n$ is acyclic for any strict order $\prec$ on $P_1, \ldots, P_n$.*

The last Proposition says that when each one of the given preference relations is acyclic, the necessary and sufficient conditions for the acyclicity of Pr_r established by Proposition 4 are always met. This can be verified by considering that under the circumstances, every cycle in $P_\cup$ involves at least two preference relations, one of which has necessarily a lower priority than the other; thus, any arc (x, y) in the cycle coming from a preference relation with a lower priority can play the role of (o_{u-1}, o_u) in Proposition 4.

The last Proposition shows that in presence of a cyclic union, the RPR rule produces an acyclic relation. However, the rule turns out to be too restrictive in that it may exclude from the result preferences that are unproblematic (*i.e.*, preferences which do not produce any cycle if included in the result). In other words, RPR may not produce a maximum acyclic sub-graph of $P_\cup$.

Consider for instance the following preference relations: $P_1 = \{(2,3)\}$, $P_2 = \{(1,2)\}$ and $P_3 = \{(3,2)\}$, together with the following strict order: $P_1 \prec P_2 \prec P_3$. In this case, we have $Pr_r = \{(2,3)\}$, which is a non-maximum acyclic sub-graph of $P_\cup$, since it misses the preference $(1,2)$.

Furthermore, there may be maximum acyclic sub-graphs of $P_\cup$ that RPR cannot define under any $\prec$. The example presented in Figure 2 is a case in point: we have $Pr_r = P_1$ (if $P_1 \prec P_2$) or $Pr_r = P_2$ (if $P_2 \prec P_1$). But for no ordering of the preference relations we can have $Pr_r = R_1$ or $Pr_r = R_2$ (also shown in Figure 2).

Let us now consider Pr_u. It is easy to construct an example that falsifies the condition of Proposition 5 (*i.e.*, a cyclic $P_\cup$ leading to a cyclic Pr_u): Assume $P_1 \prec P_2$ and $P_1 = \{(1,2)\}$ whereas $P_2 = \{(2,3),(3,1)\}$. In this case, none of the arcs making up the cycle $(1,2,3,1)$ in $P_\cup$ has a reverse arc as required by Proposition 5. In fact, $Pr_u = P_\cup$ and so we have a cyclic Pr_u.

For Pareto, the situation is identical.

COROLLARY 1. *The Restricted Pareto Pa_r of n acyclic preference relations $P_1, \ldots, P_n$ is acyclic.*

As for Proposition 8, the acyclicity of each preference relation P_i implies that the condition established by Proposition 6 are always met. In particular, each cycle in $P_\cup$ comes from preferences belonging to different relations; then, any arc (x, y) in the cycle is missed at least by one P_i, *i.e.* $x \overline{P_i} y$, and as such it plays the role of (o_{u-1}, o_u) in Proposition 6.

Since Pa_r is a subset of Pr_r (Proposition 2), we have that the above remarks related to the completeness of Pr_r carry over to Pa_r.

Finally, the previous example can be used also to show that a cyclic $P_\cup$ may lead to a cyclic Pa_u.

We may then conclude that when each preference relation is acyclic but their union is cyclic, both Pr_r and Pa_r are acyclic but incomplete. On the other hand, their unrestricted versions Pr_u and Pa_u may be cyclic.

4.3 Cyclic preference relations

In considering cyclic preference relations, we distinguish between two cases:

- the cycle involves only two objects.

  We call these objects *equivalent* in the sense defined earlier.

- the cycle involves at least three objects.

In the classical methods, object equivalence arises only in very special circumstances or not at all, as shown by the following corollary (that follows from Propositions 4 to 7):

COROLLARY 2. *Let $P_1, \ldots, P_n$, be n preference relations and let $\prec$ be any strict ordering on them. Then, for any two objects x, y*

- *$x Pr_r^{\equiv} y$ iff $x P_i^{\equiv} y$ for all $i = 1, \ldots, n$, and*
- *$x Pr_u^{\equiv} y$ iff the following hold:*
  1. *for some i we have $x P_i^{\equiv} y$, and*
  2. *for all $j = 1, \ldots, n$ we have either $x P_j^{\equiv} y$ or $x P_j^{\#} y$.*

Moreover, both Pa_r and Pa_u are asymmetric.

Thus, two objects are equivalent with respect to Restricted Prioritized if they are equivalent with respect to each preference relation. In this case they are incomparable for both versions of Pareto. For Unrestricted Prioritized, two objects are equivalent if they are comparable on some dimension but in no dimension one of the two is strictly preferred to the other. Pareto rules out object equivalence by being asymmetric in both its variants.

For cycles involving three or more objects, the situation is much worse. If some of the $P_1, \ldots, P_n$ contain such a cycle, then all variants of Prioritized and Pareto may produce a cyclic result.

The following example shows that this is indeed the case for RPA. Suppose $P_1 = P_2 = \{(1,2),(2,3),(3,1)\}$. Then, it is easy to see that $Pa_r = P_1$. Notice that if we assume transitivity of the P_i's, then for all objects x and y involved in a cycle, we have $x P_i^{\equiv} y$ and therefore $(x, y) \notin Pa_r$. In the example, transitivity of the P_i's implies that $P_i^< = \emptyset$ for all i and consequently $Pa_r = \emptyset$.

Since Pa_r is the smallest relation among those produced by the classical methods, we have that all methods can produce a cyclic result.

5. CONCLUDING REMARKS

From an information system perspective, there are two basic requirements that must be satisfied when combining preference relations into a single preference relation: completeness and consistency. Completeness means that the result must be the union of all preference relations, while consistency means that the result must be an acyclic relation.

However, it is rarely possible to satisfy these two requirements simultaneously, due to the fact that the union of the given preference relations may be cyclic. Given the importance of acyclicity in ranking the objects, we have argued that a reasonable compromise is to aim at a maximum acyclic sub-graph of the union of the given relations.

We have analyzed two classical methods for combining preference relations, Prioritized and Pareto, each in two variants: restricted and unrestricted. We have shown that all four methods are inadequate with respect to the above requirements, as they may produce a result that is either incomplete or cyclic. In particular,

- In the (fully unproblematic) case when the union of the given preference relations is acyclic, both restricted approaches are incomplete, while the unrestricted ones are optimal (*i.e.*, complete and acyclic).
- When each preference relation is acyclic but their union is cyclic, both restricted approaches produce an acyclic result, which however looses more preferences than needed to obtain acyclicity; under the same circumstances, the unrestricted methods may produce a cyclic result.
- Finally, when one or more preference relations have cycles involving at least three objects, all four methods may produce a cyclic result.

Thus, all four classical methods are for various reasons unsatisfactory. However, if we look at the problem from a purely computational perspective, we are faced with the fact that computing a maximum acyclic sub-graph of a given graph is known to be NP-hard. This fact sheds a different light on the classical approaches.

In fact we may conclude that both Prioritized and Pareto trade off efficiency to optimality (unless P=NP). In their unrestricted variants, both methods achieve optimality only when it is computationally easy to do so, (*i.e.*, when the union is acyclic). In all other cases, they retain efficiency while loosing optimality.

6. REFERENCES

[1] R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. In *Proc. of ACM SIGMOD*, pages 297–306, Dallas, USA, 2000.

[2] H. Andreka, M. Ryan, and P.-Y. Schobbens. Operators and laws for combining preference relations. *Journal of Logic and Computation*, 12(1):13–53, 2002.

[3] R.I. Brafman and C. Domshlak. Preference handling - an introductory tutorial. *AI Magasine*, 30(1), 2009.

[4] J. Chomicki. Querying with intrinsic preferences. In *Proceedings of the 8th International Conference on EDBT*, pages 34–51, Prague, Czech Rep., 2002.

[5] J. Chomicki. Semantic optimization of preference queries. In *Proc. of 1st Int. Sym. on Appl. of Constraint Databases*, number 3074 in LNCS, 2004.

[6] J. Chomicki. Iterative modification and incremental evaluation of preference queries. In *Proc. of FoIKS*, pages 63–82, 2006.

[7] Jan Chomicki. Preference formulas in relational queries. *ACM Trans. Database Systems*, 28(4):427–466, 2003.

[8] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms*. The MIT Press, 2nd edition, 2001.

[9] Peter C. Fishburn. Nontransitive preferences in decision theory. *Journal of Risk and Uncertainty*, 4(2):113–124, April 1991.

[10] Oswald Huber. Nontransitive multidimensional preferences: Theoretical analysis of a model. *Theory and Decision*, 19:147–165, January 1979.

[11] J. Kacprzyk and M. Roubens, editors. *Non-conventional preference relations in decision making*. Springer-Verlag, 1988.

[12] W. Kiessling and G. Köstler. Preference SQL design, implementation, experiences. In *In Proceedings of 28th International Conference on Very Large Data Bases*, pages 990–1001, Hong Kong, China, 2002.

[13] Werner Kiessling. Foundations of preferences in database systems. In *Proc. of VLDB*, pages 311–322, 2002.

[14] Georgia Koutrika and Yannis E. Ioannidis. Constrained optimalities in query personalization. In *Proc. of ACM SIGMOD*, pages 73–84, 2005.

[15] Georgia Koutrika and Yannis E. Ioannidis. Personalized queries under a generalized preference model. In *Proc. of ICDE*, pages 841–852, 2005.

[16] Alantha Newman. Approximatng the maximum acyclic subgraph. Master's thesis, Massachussets Institute of Technology, Dept. of Electrical Engineering and Computer Science, 2000.

[17] E.M. Nguer and N. Spyratos. A user-friendly interface for evaluating preference queries over tabular data. In *Proc. of SIGDOC08, the 26th ACM International Conference on Design of Communication*, Lisbon, Portugal, Sept 2008.

[18] M. Öztürk, A. Tsoukiàs, and Ph. Vincke. Preference modelling. In J. Figueira M. Ehrgott, S. Greco, editor, *Multiple Criteria Decision Analysis: State of the Art Surveys,*, pages 27–73. Springer Verlag, 2005.

[19] P.Georgiadis, I.Kapantaidakis, M.Nguer, N.Spyratos, and V.Christophides. Efficient rewriting algorithms for preference queries. In *Proc. of ICDE08, the 24th International Conference on Data Engineering*, Cancun, Mexico, April 2008.

[20] Amartya Sen. Social choice theory. In K. J. Arrow and M.D. Intriligator, editors, *Handbook of Mathematical Economics*, volume 3 of *Handbook of Mathematical Economics*, chapter 22, pages 1073–1181. Elsevier, September 2005.

[21] N. Spyratos and V. Christophides. Querying with preferences in a digital library. In *Dagstuhl Seminar (N 05182), Federation over the Web*, number 3847 in LNAI, May 2005.

[22] Tversky. Intransitivity of preferences. *Psychological review*, 76(1), 1969.

Personalization through Query Explanation and Document Adaptation

Anthony Ventresque[1*], Sylvie Cazalens[2], Thomas Cerqueus[2],
Philippe Lamarre[2] and Gabriella Pasi[3]
[1]SCE, Nanyang Technological University, Singapore
[2]LINA, Université de Nantes, France
[3]DiSCo, Università di Milano Bicocca, Italy
[1]aventresque@ntu.edu.sg, [2]FirstName.LastName@univ-nantes.fr, [3]pasi@disco.unimib.it

ABSTRACT

In this paper a new formal approach is proposed to retrieval personalization, based on both a query personalization process at the client's side and a light document adaptation at the information server's side. The proposed solution relies on the use of a domain ontology: queries and documents are in fact indexed by sets of concepts. The query personalization process is finalised to clarify what we call the central query concepts based on the importance of linked concepts in the considered ontology. The initial query as well as its clarifications are sent to the server, which revises the document representations based on both the query and the concepts clarifications. The proposed solution does not require that the information server maintains any user profile.

Keywords

Query Explanation, Document Adaptation, Similarity and Propagation, Semantic Vector Space

1. INTRODUCTION

Personalization is nowadays an important issue for many data and information retrieval applications, aimed at enhancing the user experience and business profits. Coping with a huge and increasing amount of data accessible from the Web, retrieval systems need to display not only information relevant to a specific query, but also information that match specific users needs, interests, preferences. And this is seen as an important marketing tool and a requirement for many electronic businesses.

Most personalization models are based on two important and complementary aspects: (i) implicit or explicit collection and consequent representation of user's behavior, preferences, interests; (ii) leveraging that knowledge during the retrieval process. This is mainly done by expanding queries [10], by re-ranking search results or by re-indexing documents [2]. Some of these tasks, like expanding queries, can be achieved either at the user's side or at the information providers side (i.e. server's side). This paper presents a personalized information retrieval approach which does not assume any user profiling by the information server. Our model favors (semi-)automatic query clarification at the user's side. Based on the query clarification process, the server reconsiders the document representations in the light of both the query and its clarification, thus enhancing the query evaluation process by specifically adapting it to the user. This approach can be useful when the server privacy policy commits it to not profiling the user.

Besides an ever increasing amount of information, these last ten years have witnessed great research interest in semantics, in particular with the definition of many domain ontologies (like in medecine, biology, almost any domain) and linked technologies. Our approach relies on the usage of a domain ontology, with queries and documents (both semi-structured or unstructured) indexed by its set of concepts as semantic vectors [1]. Each concept is weighted according to its representativeness of the document (respectively the query). Relevance is modeled as the closeness of the two vectors, as in the classical vector space model.

This paper does not focus on indexing, as we assume that it is performed on the serve's side. Given a user query (user's side) and document vectors (server's side), the objectives of our approach are to define (i) a query clarification process and (ii) a light weight adaptation process to tune the document representations to the query without requiring re-indexing.The aim of the whole approach is to conceptually enhance the query evaluation process. The proposed solution relies on several assumptions and choices.

First, each weighted concept of the query is explained separately. This seems more precise to us than a classical query expansion, which may lead to an unbalanced role of the original components of the query [10].

Second, the explanation of a given concept considers two notions: given the user's domain ontology, we assume that a similarity function on the set of concepts specifies to which extent a given concept is similar to another one. This is part of the user's modeling of the domain. However, to our view, the use of a static similarity measure is not enough to express the importance of a concept linked to a query.

*Anthony Ventresque's research is supported by A*Star SERC grant number 072 134 0055.

VLDB '10, September 13-17, 2010, Singapore

Indeed, two different search contexts may require to give more or less importance (interest) to a same similar concept. We formalize this intuition by introducing the concept of propagation function.

Both the similarity and the concept importance values are automatically computed, but the user can keep control on the process. As the propagation of importance may vary depending on the search context, the user should have a toolbox with several propagation functions which s/he can choose or which are automatically proposed depending on the context (there may be a profiling module at the users side which helps). Finally, both the initial query and the concept explanations (which are vectors) are sent to the information provider, which has to evaluate the relevance of the stored documents with respect to the query. As previously explained, our choice is to avoid re-indexing. Then, given the document representations (i.e. vectors), there are two options: (i) comparing each concept explanation with each document, and then aggregate the results to get a global relevance measure or (ii) producing an adapted document representation (without changing the stored one) which better characterizes each document with respect to the search needs expressed by the concept explanations. We have chosen the second option, in which, at the end, the relevance computation considers the initial query and the document adaptations.

Our contributions are: (i) a new formal approach to personalization which encompasses a query personalization process together with a light document adaptation; (ii) not to require that the document provider maintains user profiles; (iii) a non-intrusive solution for existing systems, as it can be plugged in systems without need of document reindexing, query reformulation nor new matching function.

In the remaining of this paper, we first present a motivating example which shows that questions "how concepts are similar?" and "how much of them are interesting and to what extent?" are crucial for personalized retrieval; this is the core of our solution, and hence we present its architecture (Section 2). Then we formally define query personalization (Section 3) and document adaptation (Section 4). In Section 5 we give a cost analysis and validate the personalization of our solution. And after a discussion on the main assumptions of our work (namely ontological heterogeneity, and similarity and propagation functions) in Section 6, some conclusions are sketched (Section 7).

2. MOTIVATING EXAMPLE AND SYSTEM ARCHITECTURE

While selecting query terms that are fully compatible to documents' providers terms (index terms) is in itself a difficult problem, a same term could also have slightly different meanings to different users (term ambiguity). For instance let us assume three users Alice, Bob and Chikako, willing to adopt a dog. Their request, e.g. "I would like a dog", is very straightforward, and they can send it to an animal welfare organization nearby their living place. Unfortunately, there are a lot of sheltered dogs in these organizations, and scanning their data base could be tedious. Hence, it should be useful to specify which kind of dog each people intend to adopt. On the other hand, if they want a pedigree dog, they could be disappointed when using the keyword "dog". Indeed, dog breeders are canine specialists, and their animals shall not be deemed to be "dog", but "labrador", "akita", or whatever. In both cases, a more accurate description of the user's preferences, i.e. the intended objects, should be useful to specify or to expand queries.

Let's Alice and Bob more likely consider as a dog prototype the labrador, while Chikako's dog prototype is represented by akitas (see Figure 1). Even if the concept *dog* has the same meaning in their mind, the descendant (more specific) concepts of dogs are not all similar, being some are more relevant than others. As a consequence when querying the animal welfare organization data base, they do not look for the exactly same items. Then, a solution to improve the results expected from the evaluation of their queries should take into account the users' prototypal concepts and their similar concepts. This means to consider a central concept (the prototype) and to formalise a *concept similarity function*; the user will finally decide which items in the ranked list will be relevant to her/him. Thus, while labradors are the prototype "dogs" to Alice and Bob, it may happen that dalmatians and akita are still acceptable to Alice while not to Bob. Likewise, Chikako considers that dalmatians and labradors, even if they are less relevant than akita, are still ok. We call this combination of proximity to the central concept and interest values a *propagation function*. In Figure 1 we show the interest values that Alice, Bob and Chikako give to the concepts of their ontology according to their similarity to *dog*.

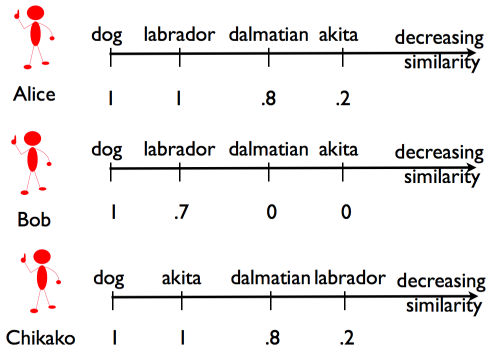


Figure 1: Alice, Bob and Chikako's similarity ranking of concepts and the propagation of their interest, both w.r.t. concept *dog*.

The propagation function aims at describing a dimension of a query, i.e. one of its main concept. As each user manages its own propagation (own similarity and own interest values), we call such a description a *personalized dimension* of the query. Once the query is personalized, our solution is to keep the query unchanged, but to transform the document representations according to the personalized dimensions. It brings us to *adapted documents*. Finally, the adapted documents are compared to the initial query. Architecture of our system is composed of five modules (Figure 2), over both user and document provider. Actually, the basic modules (white) are already provided by current systems. You can see two *semantic indexing* modules on both user's and document provider's side; these modules represent the queries and documents based on the representation model of the IR system (in our case: semantic vectors). Every system has also a *relevance computation* module (matching module), which ranks documents according to their relevance

to the query (cosine). We add to this classical architecture three new modules (grey). On the user's side, the *query personalization module* explains the central concepts of the query, according to user's similarity and propagation functions. We see below in this paper how a user could obtain these functions (see Section 6.2). The descriptions are then given to the *data adaptation* module, which transform the document representation w.r.t. them.

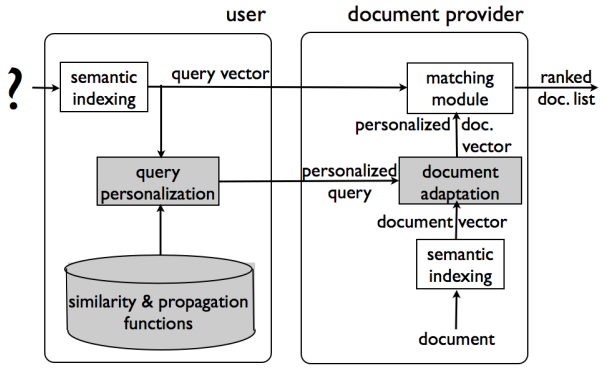


Figure 2: System architecture. All new grey modules are included in a classical semantic retrieval system.

3. QUERY PERSONALIZATION

In this section, after a description of the semantic vectors, we present a formalization of the propagation of users' interests, which constitutes the main process of query personalization. We provide some inputs on similarity and propagation functions later in this paper (see Section 6.2).

3.1 Semantic Vectors

In the vector space model [1], both queries and documents are represented as vectors of keywords (terms). If there are n keywords, each query or document is represented by a vector in the $n-$dimensional space. Relevance of a document to a query can then be calculated by measuring the proximity of the two vectors. An approach based on *semantic vectors* [16] uses the same kind of multi-dimensional linear space except that it no longer considers as dimensions the keywords, but *concepts* belonging to a considered ontology: the content of each query (respectively document) is represented by a semantic vector according to each concept.

We consider a very general definition of ontology [6]: it is a set of concepts together with a set of relations between those concepts. The only assumption we make is to be able to compute a similarity between concepts of an ontology, whatever the relations used are. In the rest of the paper, we assume the existence of an ontology Ω, $\mathcal{C}_\Omega$ being its set of concepts. Then, a simple formal definition of a semantic vector can be the following:

Definition 1 (Semantic Vector). *A* semantic vector $\overrightarrow{v_\Omega}$ *is an application defined on the set of concepts* $\mathcal{C}_\Omega$ *of the ontology:*

$$\forall c \in \mathcal{C}_\Omega, \overrightarrow{v_\Omega} : c \rightarrow [0,1]$$

Reference to the ontology will be omitted whenever there is no ambiguity.

3.2 Propagation of Interest

Conceptual similarity is a function centered on a concept: it gives a value to every concept according to its similarity to the central concept.

Definition 2 (Similarity Function). *Let* c *be a concept of* $\mathcal{C}_\Omega$. sim_c: $\mathcal{C}_\Omega \rightarrow [0,1]$, *is a similarity function iff* $sim_c(c) = 1$ *and* $0 \leq sim_c(c_j) \leq 1$ *for all* $c_j \neq c$ *in* $\mathcal{C}_\Omega$.

Given a similarity function and a central concept c, we define a *propagation function* as a function which describes the importance of every concept according to c. We assume this function is monotonically decreasing.

Definition 3 (Propagation function). *Let* c *be a concept of* $\mathcal{C}_\Omega$*; and let* sim_c *be a similarity function. A function* $\mathcal{P}f_c$: $[0,1] \mapsto [0,1]$, $sim_c(c') \rightarrow \mathcal{P}f_c(sim_c(c'))$ *is a* propagation function from c *iff* $\mathcal{P}f_c(sim_c(c)) = 1$, *and* $\forall c_k, c_l \in \mathcal{C}_\Omega\ sim_c(c_k) \leq sim_c(c_l) \Rightarrow \mathcal{P}f_c(sim_c(c_k)) \leq \mathcal{P}f_c(sim_c(c_l))$

We have suggested some propagation functions in [14]. They are inspired by membership functions used in fuzzy logic [18], i.e. the most similar concepts are given the value 1, the most dissimilar are weighted with 0, and in between, concepts receive a value according to their similarity. It is defined by two parameters l_1 (length of the interval where concepts have weight 1) and l_2 (length of the interval where concepts have non zero weight) such that:

$$\mathcal{P}f_c(x) = f_{l_1,l_2}(x) = \begin{cases} 1 & if\ x \geq l_1 \\ \frac{1}{l_1 - l_2}x + \frac{l_2}{l_1 - l_2} & if\ l_1 > x > l_2 \\ 0 & if\ l_2 \geq x \end{cases}$$

3.3 Semantic Personalized Query

As we said in the introduction we do not expand (by propagation) in a single new vector the weights of the central concepts of the query. Moreover, each central concept c of a query $\overrightarrow{q}$ is personalized in a separate vector. Thus a personalized dimension $\overrightarrow{persD_c}$ is a semantic vector which records the propagation of one concept only. Let $\mathcal{C}_{\overrightarrow{q}}$ be the set of central concepts, i.e. important concepts for the query: e.g. any weighted concept, a concept weighted with a greater value than a threshold, etc. $\mathcal{C}_{\overrightarrow{q}} = \{c : c \text{ is a central concept}\}$.

Definition 4 (personalized dimension). *Let* $\overrightarrow{q}$ *be a query vector and let* c *be a concept in* $\mathcal{C}_{\overrightarrow{q}}$. *A semantic vector* $\overrightarrow{persD_c}$ *is a* personalized dimension, *iff* $\exists \mathcal{P}f_c \forall c' \in \mathcal{C}_\Omega, \overrightarrow{persD_c}[c'] = \mathcal{P}f_c(sim_c(c'))$.

The mathematical expression ending the definition means that no matter how the $\overrightarrow{persD_c}$ is obtained the only restriction is that no concept can have a greater weight than c, which is *always* 1, and not the original value, because a personalized dimension is an explanation of a dimension of the query. The original query, and then the original values of the central concepts, are kept for the matching process.

A personalized query is the set of personalized dimensions, one for each central concept of a query: $persQ_{\overrightarrow{q}} = \{\overrightarrow{persD_c} : c \in \mathcal{C}_{\overrightarrow{q}}\}$. Figure 3 shows the personalization of a query $\overrightarrow{q}$ with two weighted concepts c_4 and c_7.

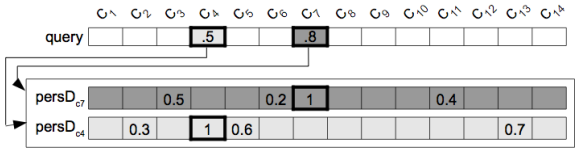


Figure 3: A personalized query composed of 2 personalized dimensions.

4. DOCUMENT ADAPTATION

Once user has explained central concepts of his/her query $\overrightarrow{q}$, s/he sends the personalized query to the document provider, who adapts his/her documents to the $persQ_{\overrightarrow{q}}$. Adaptation is not a reindexing of the documents, like with Bordogna and Pasi [2] for instance. It is a lightweight filter of documents through what the query explains as important in its personalized dimensions; and it results in a new vector, $\overrightarrow{persR_{\overrightarrow{q}}^{\overrightarrow{d}}}$ (simplified into $\overrightarrow{persR}$). If a concept c_i is relevant for $\overrightarrow{persD}_{c_j}$, then every document with this concept c_i should give that information in its adaptation. In fact, for any $\overrightarrow{persD}_{c_j}$, documents retain a unique value in the adaptation vector for concept c_j, which is the best correlation between personalization value of $c_i \in \overrightarrow{persD}_{c_j}$ and value $\overrightarrow{d}[c_i]$. While all concepts involved in some personalized dimension are already captured by this process, their values are then null in the adaptation. Other concepts, not relevant for any personalized dimension, keep their original value, as they show some dimensions of the document not relevant for the personalized query. Indeed, the norm of the vector gets higher (and consequently, its relevance lower). For example, this is the case for concepts c_1 and c_9 in Figure 4.

Algorithm 1 details the computation of the personalization of document representation $\overrightarrow{d}$.This algorithm ensures that all the central concepts of the initial query vector are also weighted in the personalized document representation as far as it is related to them. W.r.t. the query, the personalization of the document representation is more accurate because it somewhat enforces the characterization of the document over each dimension of the query.

The example of Figure 4 illustrates the computation of a $\overrightarrow{persR}$. Each $\overrightarrow{persD}$ of the personalized query is combined with the semantic vector of the document. Let us consider $\overrightarrow{persD}_{c_4}$. In the document, the weight of c_4 is null. However, the personalized dimension related to c_4 weights other concepts. In particular, we have $\overrightarrow{persD}_{c_4}[\ c_2\ ] = 0.3$. As $\overrightarrow{d}[\ c_2\ ] = 1$, the resulting product is 0.3. This value improves $\overrightarrow{d}[\ c_4\ ]$ (which is null), so we keep it in the adaptation of the document representation. Hence, in the $\overrightarrow{persR}$, we can express that the document is related to concept c_4 of the query, even if it wasn't the case initially. Likewise, three concepts of the document (c_6, c_7 and c_{11}) are important to $\overrightarrow{persD}_{c_7}$, and the adaptation retains only one value for $\overrightarrow{persD}_{c_7}[c_7] = 0.6$. Note that while a classical expanded query would have given one single value from central concept c_4 (but possibly on c_2 instead of c_4), expansion should have weighted 3 concepts from c_7 (c_6, c_7 and c_{11}). Our solution does not add as much noise as it could with classical expansion. Concepts c_1 and c_9 eventually keep their original value in $\overrightarrow{persR}$ of document $\vec{d}$ because they are not involved in any $\overrightarrow{persD}$.

Algorithm 1: Adaptation of document representation wrt. a query.

input : a semantic vector $\overrightarrow{d}$ and a personalized query $persQ_{\overrightarrow{q}}$ on an ontology Ω
output: a semantic vector $\overrightarrow{persR_{\overrightarrow{q}}^{\overrightarrow{d}}}$.
begin
  forall $c \in \mathcal{C}_{\overrightarrow{q}}$ **do**
    forall $c' : \overrightarrow{persD}_c[c'] \neq 0$ **do**
      $\overrightarrow{persR_{\overrightarrow{q}}^{\overrightarrow{d}}}[c] \leftarrow$
      $\max(\overrightarrow{d}[c'] \times \overrightarrow{persD}_c[c'], \overrightarrow{persR_{\overrightarrow{q}}^{\overrightarrow{d}}}[c])$;
  forall $c \notin \mathcal{C}_{\overrightarrow{q}}$ **do**
    if $\exists c' \in \mathcal{C}_{\overrightarrow{q}} : \overrightarrow{persD}_{c'}[c] \neq 0$ **then**
      $\overrightarrow{persR_{\overrightarrow{q}}^{\overrightarrow{d}}}[c] \leftarrow 0$
    else
      $\overrightarrow{persR_{\overrightarrow{q}}^{\overrightarrow{d}}}[c] \leftarrow \overrightarrow{d}[c]$;
  return $\overrightarrow{i}_d$;
end

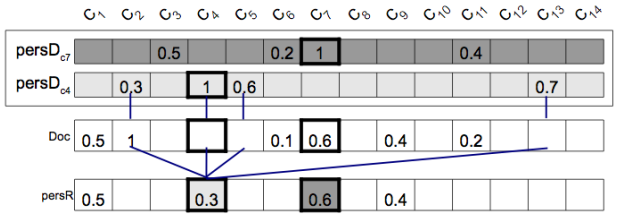


Figure 4: Obtaining the adapted document representation.

5. EXPERIMENTS

Our goal is to validate our approach through several steps: first step is cost analysis which enables to quantify the additional costs induced by the method. Second one is just to verify that given a query and different search contexts, users get different results; this is can be viewed as a minimum requirement to get personnalized results. Finally, the method should be faced with a significant number of users who would estimate whether (or to which extent) they get personalized results. As we currently judge our number of users not significant enough, the paper focuses on the first two steps.

Complexity of our solution relies on the complexity of similarity and propagation functions, which together define query personalization, and document adaptation. There exist a lot of similarity measures (see Section 6.2) but they always consist on two nested loops. Let n be the number of concepts in the ontology, then similarity computation is in $O(n^2)$. Propagation gives a weight to every concept; and there are as many propagation functions as there are central

concepts. Assume m the number of central concepts, then we need $O(m \times n)$ to compute propagation. As m is generally very small compared to n, query personalization has a complexity of $O(n^2 + n)$. However, these two steps can be computed before and cached; so it is not always needed to compute them whenever user queries the system. Adaptation mainly consists of a loop on every concept of the ontology for every document and $\overrightarrow{persD}$. Then, for every central concept and every document, values are inserted at indices of central concepts. Finally, a loop is processed on concepts to put the values of concepts not involved in any $\overrightarrow{persD}$. If d is the number of documents, we have an adaptation computation in $O(d(2(m \times n) + m \times log(n) + n))$. But this can be strongly reduced by using proper data model: big vectors (as many indices as there are concepts in the ontology) are useless while documents and queries are not expressed on all the concepts but a very small subset, etc. The worst case is not realistic and we assume a fast computation, by replacing at least n by n' which is drastically smaller. For instance, we shown in [14] that a good propagation impacts on 25 concepts for each central concept.

Our experiments use the Cranfield corpus and WordNet (considered as a "lightweight" ontology) to index the documents. We developped a prototype software called *Mysins* [15] with a service oriented architecture. In *Mysins*, search can be personalized by choosing the similarity and propagation functions. The server side of Mysins runs the document adaptation module. We ran the 225 queries of the corpus. The number of retrieved documents is 50 (among the 1400 of the corpus). This later assumption seems reasonable as several user behavior analysis show that users generally consult the first result pages only. We use two different similarity measures: Wu and Palmer [17] noted sim_1, and a modified version of sim_1 (which permutes values of three highest similarity values) as sim_2. Likewise we use two different propagation functions: $prop_1 = f_{0.95,0.9}$ and $prop_2 = f_{0.8,0.6}$ (see Equation 3.2).

In order to compare two sets of retrieved documents with their relevance values, we consider two measures. First one (Jaccard coefficient) measures the similarity of the two sets of documents (without considering their relevance value). It is defined as the number of documents in the intersection divided by the number of documents in the union of the two sets. Second one takes into account the order in the ranking of retrieved documents. We have chosen a modified version of Rank Distance (RD) [4]: each document is given a value according to its position in the top-50, 50 for the 1st, 49 for the 2nd, etc. and 0 for the 51st onwards. Value of each document in first list is then compared to its value in second list. This measure gives of course more importance to permutations on top of the list of retrieved documents. You can see in Figure 5 (a), (b) and (c) the results for $\langle(sim_1, prop_1), (sim_1, prop_2)\rangle$, $\langle(sim_1, prop_1), (sim_2, prop_1)\rangle$ and $\langle(sim_1, prop_1), (sim_2, prop_2)\rangle$ respectively. Each dot corresponds to the comparison of answer lists of a query, using the given parameters. X-axis shows Jaccard measure and y-axis the home-made RD.

It is first worth noticing that dots are not close to (0,0), which means that there are differences between the two results sets. Most dots have Jaccard values between 0.05 and 0.4, while their RD values are between 0.1 and 0.6. This means that results sets are different (not the same collection of documents) and their ranking are even more different. Propagation eventually seems more important than similarity, because Figures 5 (a) and (c) have more scattered dots, with higher dissimilarity and/or disorder values in average.

This section has proven that: (i) additional cost of our solution is limited and (ii) in different contexts the results sets are different and show a personalization of the retrieval. Future work intends to validate the approach with "real" users, and their satisfaction.

6. DISCUSSION AND RELATED WORK

In this section, we first position our assumptions and propositions w.r.t. related work. We then discuss how collection of similarity and propagation functions can be thought.

6.1 IR, Personalization and Ontologies

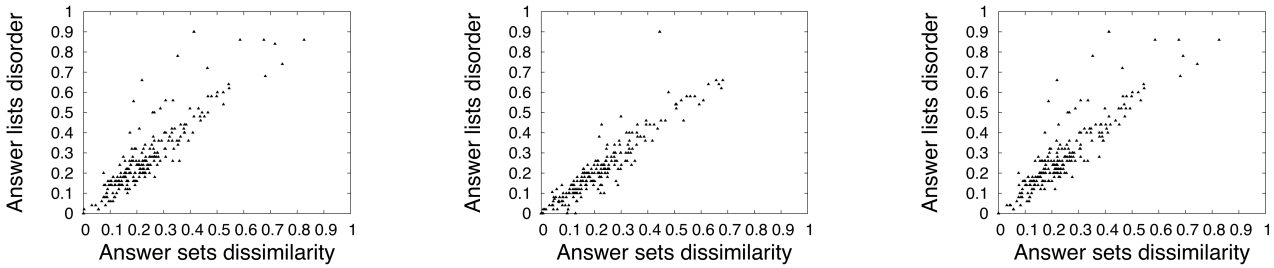
Context formalization for IR has focused a lot of attention in past few years [9]. Many work address this problem through the construction of a contextual space, collecting information on past queries, users clicks, etc. While most of them use terms to characterize the context, Mylonas et alii [9] propose to use an ontology. Their work is very interesting and can be compared to ours. But it does not use semantic vectors and our solution is more lightweight.

Query expansion has been seen promising to enhance small-size queries in order to help IR engines [3]. But while query expansion is a worthwhile contribution to IR, offering more relevant results, it often adds noise in the retrieval. So IR systems need to know when to use it [12]. Our solution do not use a query expansion, but a description of central concepts of the query through a propagation function on the concepts of the ontology. We have proven in [14] that our solution performs better than expansion in general case. And it is specifically more resistant to the use of many concepts.

Assuming a total agreement on ontologies on both sides is not realistic: an ontology is a conceptualization of knowledge upon the world, and we can hardly imagine a unique model of the world for every users. Alignment of ontologies, i.e. mappings between parts of the ontologies [5], are mostly used to address these problems. However these alignments are often incomplete: either because the process is time or resource consuming, or because users do not want to share all their conceptualizations, or because it is not always possible. While query or document indexing could be done on unshared parts of ontologies, it is useless. Indeed, every unshared element could not be understood, and hence no document would be relevant (cosine works only on common parts of queries and documents vectors). However, these unshared parts are meaningful for users and document providers and they are worthy of being used. We propose in [14] an *interpretation* process which let users and providers free to use their own ontologies during the information retrieval process. We are still working on an extension of the system described in this paper to a heterogeneous context.

6.2 Similarity and Propagation

Similarity functions have been studied for a very long time [11, 13], etc. There exist a lot of different similarity functions, depending on the application and some desired properties. While most of them are context-independant, some takes it into account [7, 8]. Even if the problem of finding a personalized similarity function is not exactly addressed in these studies, we assume it could be done quite easily. For



(a) (b) (c)

Figure 5: Comparisons of three pairs of parameters: same similarity and different propagation functions (a), different similarity and same propagation functions (b) and different similarity and different propagation functions (c), using Jaccard (x-axis) and home-made RD (y-axis).

instance, we could imagine to collect the relative use frequency of sister concepts (e.g. *labrador* and *akita*) to give them different similarity values.

We do not address either the issue of propagation function personalization. It is a topic in itself and we focus here on the general process of personalization. However, we can imagine to first use a basic propagation, like we used in the worked mentioned before; then the system could collect feeedback from the user and change this basic propagation, according or not to some context. We would like to focus later on this issue.

7. CONCLUSION

Personalization of answering, content filtering, recommendation systems, etc. have been a topic of immense interest in recent times. While some solutions use a collection of user's behavior at providers' side or may substantially modify the retrieval system, our solution does not require that the information server maintains any user profile and is non-intrusive for retrieval systems. Moreover, we focus on a description of the query in order to watch documents in the light of its needs, and do not invent a new query formulation paradigm, or a reindexing of documents. Once users provide similarity and propagation functions, our system is lightweight and can be integrated in most information systems, assuming the system use semantic vector representations for queries and documents; then our solution can be used with documents, comments in blog, etc.

8. REFERENCES

[1] M. W. Berry, Z. Drmac, and E. R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Rev.*, 41(2):335–362, 1999.

[2] G. Bordogna and G. Pasi. Personalised indexing and retrieval of heterogeneous structured documents. *Information Retrieval*, 8(2):301–318, 2005.

[3] P.-A. Chirita, C. S. Firan, and W. Nedjl. Personalized query expansion for the web. In *SIGIR*, pages 7–14, New-York, 2007.

[4] L. P. Dinu. On the classification and aggregation of hierarchies with different constitutive elements. *Fundam. Inf.*, pages 39–50, 2002.

[5] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.

[6] A. Gómez-Pérez, M. Fernández, and O. Corcho. *Ontological Engineering*. Springer-Verlag, London, 2004.

[7] V. Kashyap and A. Sheth. Semantic and schematic similarities between database objects: a context-based approach. *The VLDB Journal*, 5(4):276–304, 1996.

[8] C. Keßler, M. Raubal, and K. Janowicz. The effect of context on semantic similarity measurement. In *OTM Workshops (2)*, pages 1274–1284, 2007.

[9] P. Mylonas, D. Vallet, P. Castells, M. Fernandez, and Y. Avrithis. Personalized informtion retrieval based on context and ontological knowledge. *The Knowledge Engineering Review*, 23(1):73–100, 2008.

[10] J.-Y. Nie and F. Jin. Integrating logical operators in query expansion invector space model. In *SIGIR workshop on Mathematical and Formal methods in Information Retrieval*, 2002.

[11] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.

[12] J. Teeman, S. T. Dumais, and D. J. Liebing. To personalize or not to personalize: Modeling queries with variations in user intent. In *SIGIR*, pages 163–170, Singapore, 2008.

[13] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, July 1977.

[14] A. Ventresque, S. Cazalens, P. Lamarre, and P. Valduriez. Improving interoperability using query interpretation in semantic vector spaces. In *ESWC*, pages 539–553, 2008.

[15] A. Ventresque, T. Cerqueus, L.-A. Celton, G. Hervouet, D. Levin, P. Lamarre, and S. Cazalens. Mysins : make your semantic information system. In *EGC*, pages 629–630, 2010.

[16] W. Woods. Conceptual indexing: A better way to organize knowledge. Technical report, Sun Microsystems Laboratories, April 1997.

[17] Z. Wu and M. Palmer. Verb semantics and lexical selection. In *ACL*, pages 133–138, Las Cruces, New Mexico, 1994.

[18] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

SQL QueRIE Recommendations: a query fragment-based approach

Javad Akbarnejad
Computer Engineering Dept.
San Jose State Univ.
San Jose, CA

Magdalini Eirinaki
Computer Engineering Dept.
San Jose State Univ.
San Jose, CA

Suju Koshy
Computer Engineering Dept.
San Jose State Univ.
San Jose, CA

Duc On
Computer Engineering Dept.
San Jose State Univ.
San Jose, CA

Neoklis Polyzotis
Computer Science Dept.
Univ. of California, Santa Cruz
Santa Cruz, CA

ABSTRACT

Relational database systems are becoming increasingly popular in the scientific community to support the interactive exploration of large volumes of data. In this scenario, users employ a query interface (typically, a web-based client) to issue a series of SQL queries that aim to analyze the data and mine it for interesting information. First-time users, however, may not have the necessary knowledge to know where to start their exploration. Other times, users may simply overlook queries that retrieve important information. In this work we describe a framework to assist non-expert users by providing personalized query recommendations. The querying behavior of the active user is represented by a set of query fragments, which are then used to identify similar query fragments in the recorded sessions of other users. The identified fragments are then transformed to interesting queries that are recommended to the active user. An experimental evaluation using real user traces shows that the generated recommendations can achieve high accuracy.

Keywords: recommender systems, collaborative filtering, relational databases, interactive exploration

1. INTRODUCTION

Relational database systems are becoming increasingly popular in the scientific community in order to provide access to large volumes of scientific data. Examples include the Genome browser[1] that hosts a genomic database, and SkyServer[2] that stores large volumes of astronomical measurements. Scientific databases are usually accessed through a web-based interface that allows users to submit SQL queries and retrieve the results. Even though users have the ability to issue complex queries over large data sets, the task of knowledge discovery remains a big challenge. Users may not know which parts of the database hold useful information, may overlook queries that retrieve relevant data, or might not have the required expertise to formulate such queries. Moreover, because of the continuously increasing size of the database, an extensive exploration of the whole database is usually very time-consuming. These factors clearly hinder data exploration and limit the benefits of using a relational database system.

To address the important problem of assisting users when exploring a database, we designed the QueRIE framework (Query Recommendations for Interactive data Exploration). QueRIE assists users by generating dynamic, personalized query recommendations in ad-hoc or form-based query environments. The idea is to provide the user with a set of SQL queries that are expected to be relevant to their information needs. The user will be able to directly submit or further refine these queries, instead of having to compose new ones.

QueRIE is built on a simple premise that is inspired by Web recommender systems: If a user A has similar querying behavior to user B, then they are likely interested in retrieving the same data. Hence, the queries of user B can serve as a guide for user A. Collaborative filtering is a well known, mature technique for realizing this idea that we can borrow from Web recommender systems, but its application to database queries presents several challenges. First, SQL is a declarative language, and hence syntactically different queries may retrieve the same data. This complicates the evaluation of similarity among users, since, contrary to the web paradigm where the similarity between two users can be expressed as the similarity between the items they visit/rate/purchase, we cannot rely directly on the SQL queries. A second important challenge is how to assign importance to the data retrieved by a user's queries, since we cannot assume an explicit rating system as in the case of the Web. Finally, the recommendations to the users have to be in the form of SQL queries, since recommending specific data items may not be very intuitive. Thus, we need to "close the loop" by first decomposing the user queries into lower-level elements in order to compute similarities and make predictions, and then map the recommended elements back to meaningful and intuitive SQL queries that users can understand or refine. All those issues make the problem of interactive database exploration very different from its web counterpart.

In our previous work [2, 9], we presented the QueRIE architecture, framework, and the application of user-based collaborative filtering using witness tuples to represent user queries. In this papers, we propose an *item-based* approach that uses *query fragments* to represent the user queries. The recorded fragments are used to identify similar query fragments in the previously recorded sessions, which are in turn "assembled" in potentially interesting

[1] http://genome.ucsc.edu/
[2] http://cas.sdss.org/

Proceedings of the VLDB Endowment, Vol. 3, No. 2

queries for the active user. We show through experimentation that the proposed method generates meaningful recommendations on real-life traces from the SkyServer database.

The rest of the paper is organized as follows: in Section 2 we review related research performed in the area of query recommendations for relational databases; in Section 3 we provide a brief overview of the QueRIE conceptual framework; in Sections 4 and 5 we present the proposed fragment-based instantiation of the conceptual framework, along with some specific implementation details concerning the queries' preprocessing; Section 6 includes some experimental results that evaluate several parameters of our framework and Section 7 concludes the paper with our plans for future work.

2. RELATED WORK

Even though the problem of generating personalized recommendations has been broadly addressed in the Web context [10], only a handful of related works exist in the database context. Some work has been done in the area of personalized recommendations for keyword or free-form query interfaces [11]. In this scenario, a user queries a database using keywords or free-form text and the personalization system recommends items of interest. Our approach is different from this scenario because it aims to assist users who query relational databases using either ad-hoc or form-based queries. Also, our framework recommends queries instead of "items" from the database. Finally, QueRIE does not require from the users to explicitly declare their preferences beforehand in order to generate recommendations.

A multidimensional query recommendation system is proposed in [3, 5, 4]. In this work the authors address the related problem of generating recommendations for data warehouses and OLAP systems. In this work, the authors propose a framework for generating Online Analytical Processing (OLAP) query recommendations for the users of a data warehouse. Although this work has some similarities to ours (for example, the challenges that need to be addressed because of the database context), the techniques and the algorithms employed in the multidimensional scenario (for example, the similarity metrics and the ranking algorithms) are very different to the ones we propose.

The necessity of a query recommendation framework is emphasized in [6], where the authors outline the architecture of a collaborative query management system targeted at large-scale, shared-data environments. As part of this architecture, they suggest that data-mining techniques can be applied to the query logs in order to generate query suggestions. The authors present a general outline of a framework for query recommendations pointing out that this is a challenging process. However, they do not provide any technical details on how such a recommendation system could be implemented.

Two very recent works propose frameworks for query recommendations using the information recorded in the query logs [13, 14]. In [13], the authors propose a query recommender system that represents the past queries using the most frequently appearing tuple values. Then, after predicting which new tuples might be of interest to the end user, they reconstruct the query that retrieves them. Contrary to our work, this approach is tuple-based. Moreover, the proposed scheme works better with relations that have discrete attribute values, contrary to scientific databases, where most attributes are numeric. The authors also propose a global ranking of the queries, based on the statistics of the database and not the query logs. Both approaches are evaluated in a preliminary empirical study, yet no discussion on scalability issues is provided. In [14], the authors propose a framework that recommends join queries. They use the data recorded in the query logs and reconstruct queries, however they assume that the end user should provide the system with some tables to be used as input and other tables to be used as output, along with the respective selection conditions. This approach clearly differs from ours in that they do not take the current user's session into consideration, neither they perform recommendations in the traditional "personalized" form (i.e. finding similarities among users or items).

In our previous work [2, 9] we defined the QueRIE conceptual framework and proposed a user-based approach that focused on the tuples touched by each query in a user's session. The system finds similarities among the current and past users based on this tuples' representation of the user sessions. A session summary predicting tuples of interest is constructed and used to identify queries recorded in the query logs that touch the same tuples. Since the proposed instantiation is based on user-based collaborative filtering, an approximation technique for accelerating the real-time calculations was also proposed. Contrary to our previous work, in this work we follow the item-based collaborative filtering approach that allows most of the calculations to be performed offline, thus enhancing the real-time performance of the system. Moreover, the queries are represented by their fragments and not the tuples they retrieve. In this way, the proposed instantiation focuses on identifying similar queries in terms of structural similarity thus capturing the semantics of the database exploration. The prototype of the QueRIE framework, incorporating both instantiations, will be presented in [1].

3. PRELIMINARIES

Users typically explore a relational database through a sequence of SQL queries. The goal of the exploration is to discover interesting information or verify a particular hypothesis. The queries are formulated based on this goal and reflect the user's overall information need. As a consequence, the queries posted by a user during one "visit" (commonly called *session*) to the database are typically correlated, in that the user formulates the next query in the sequence after having inspected the results of previous queries.

Given a user i, let $\mathcal{Q}_i$ denote the set of SQL queries that the user has posed. We model this subset of the database covered by the queries of each user as a *session summary*. This summary captures the parts of the database accessed by the user and incorporates a metric of importance for each part. Contrary to Web recommender systems, where the users are represented by the items they visit/rate/purchase, in the context of relational databases, several ways to model the session summaries exist. For instance, a crude summary may contain the names of the relations that appear in the queries of the user, and the importance of each relation can be measured as the number of queries that reference it. On the other extreme, a detailed summary may contain the actual results inspected by the user, along with an explicit rating of each result tuple. Assuming that the choice of the summary is fixed for all users, we use S_i to denote the summary for user i.

To generate recommendations, the framework computes a "predicted" summary S_0^{pred}. This summary captures the predicted degree of interest of the active user S_0 with respect to all the parts of the database, including those that the user has not explored yet, and thus serves as the "seed" for the generation of recommendations. The predicted summary is defined as follows:

$$S_0^{\text{pred}} = f(\alpha * S_0, (1-\alpha) * \{S_1, \ldots, S_h\}). \quad (1)$$

In other words, the predicted summary depends on both the active user S_0 and the summaries $S_1, \ldots, S_h$ of past users.

Contrary to Web recommender systems that rely exclusively upon the summaries of past users, we introduce a "mixing factor" $\alpha \in [0, 1]$ that determines the importance of the active user's queries as opposed to these of the past users in the computation of the predicted summary. In this way, we are able to predict queries that "expand" queries previously submitted by the user, in terms of adding slightly different clauses, parameters, or restructuring the query. Intuitively, we expect the active user to behave in a similar way, by posing queries that cover adjacent or overlapping parts of the database, in order to locate the information they are seeking. We should note, however, that the framework will also predict completely different queries as well, depending on the information recorded in the query logs.

Using S_0^{pred}, the framework constructs queries that cover the subset of the database with the highest predicted importance. In turn, these queries are presented to the user as recommendations. This step differs from the respective one in Web recommender systems since, in this case, the predicted summary is not a straightforward representation of queries. On the contrary, we need to devise algorithms that, given the predicted summary, can synthesize meaningful queries that will form the recommendation set.

Overall, our framework consists of three components: (a) Session summaries: the construction of a session summary for each user based on her past queries, (b) Recommendation seed computation: the computation of a predicted summary S_0^{pred} that serves as the seed, and (c) Generation of query recommendations: the generation of queries based on S_0^{pred}. An interesting point is that components (a) and (c) form a closed loop, going from queries to session summaries and back. Again, this design choice follows the fact that all user interaction with a relational database occurs through declarative queries.

In what follows, we investigate a query fragments-based approach to modeling the queries, and consequently the users.

4. FRAGMENT-BASED RECOMMENDATIONS

In order to generate recommendations, we follow a methodology similar to the item-based collaborative filtering. This approach is based on the pair-wise similarity among the items involved in the recorded user sessions. Items that co-appear in many sessions are considered similar to each other and these similarities are used in order to generate recommendations for an active session. Contrary to user-based collaborative filtering, this technique allows the calculation of all similarities offline, thus accelerating the real-time calculations and enabling fast recommendations' generation.

In this paper, we represent each user session by the query fragments (attributes, tables, joins and predicates) identified in the respective queries. The objective is to identify fragments that co-appear in several queries posed by different users, and use them in the recommendation process. Thus, QueRIE first calculates offline the pair-wise similarities of all query fragments recorded in the query logs. These similarities are subsequently used to predict, in real time, the "ranking" (i.e. importance) of each fragment with regards to the current user session. In turn, the highest ranked query fragments are selected and used to retrieve queries that include them, which are used as recommendations. The proposed algorithm is presented in more detail in what follows.

4.1 Session summaries.

The session summary vector S_i for a user i consists of all the query fragments ϕ of the user's past queries. Let $\mathcal{Q}_i$ represent the set of queries posed by user i during a session and F represent the set of all distinct query fragments recorded in the query logs. We assume that the vector S_Q represents a single query $Q \in \mathcal{Q}_i$. For a given fragment $\phi \in F$, we define $S_Q[\phi]$ as a binary variable that represents the presence or absence of ϕ in a query Q. Then $S_i[\phi]$ represents the importance of ϕ in session S_i.

We propose two different weighting schemes for computing the fragment weights in S_i:

Binary scheme.

$$S_i = \bigvee_{Q \in \mathcal{Q}_i} S_Q. \tag{2}$$

In this scheme all participating fragments receive the same importance weight, regardless of whether they appear in many queries in the session or only one.

Weighted scheme.

$$S_i = \sum_{Q \in \mathcal{Q}_i} S_Q. \tag{3}$$

In this approach fragments that appear more than once in a user session will receive higher weight than others.

4.2 Recommendation seed computation.

Using the session summaries of the past users and a vector similarity metric, we construct the $(|F| \times |F|)$ *fragment-fragment matrix* that contains all similarities $sim(\rho, \phi)$, $\rho, \phi \in F$. Intuitively, and according to the item-based collaborative filtering approach, the more the sessions that include both fragments, the more similar these fragments are. The similarity metric employed depends on the weighting scheme that was chosen in the previous step, thus we employ Jaccard's coefficient and cosine similarity for the binary and weighted schemes respectively. We should note that all pair-wise similarities are calculated and stored off-line. This results in a very efficient execution of the algorithm in terms of computational time.

The recommendation seed, modeled by S_0^{pred}, represents the estimated importance of each query fragment with regard to the active user's behavior S_0. Similarly to the item-to-item collaborative filtering approach of web recommender systems, we employ the fragment-to-fragment similarities that are computed in the previous step:

$$S_0^{\text{pred}}[\phi] = \frac{\sum_{\rho \in R} S_0[\rho] * sim(\rho, \phi)}{\sum_{\rho \in R} sim(\rho, \phi)}, \tag{4}$$

where R represents the set of top-k similar query fragments ($k \leq |F|$). Please note that, even if we follow the binary approach, S_0^{pred} is not a binary vector.

As shown in Equation 1, we introduce a "mixing factor" $\alpha \in [0, 1]$ that allows us to include or exclude the fragments of the active user session in the recommendation process. α is a parameter of the QueRIE framework. When $\alpha = 0$ we follow the classic item-based collaborative filtering approach, whereas when $\alpha = 1$ we follow a content-based approach, in that only the fragments included in the active user's queries are taken into consideration. More discussion on the effect of α is included in Section 3.

4.3 Generation of query recommendations.

The recommendation set will include queries that have been previously recorded in the query logs. In this way, we ensure that the queries are understandable and executable, since they were authored by humans. This decision allows for faster and more intuitive recommendations, as compared to the option of synthesizing queries on the fly.

Once the predicted summary S_0^{pred} has been computed, the top-n fragments F_n (i.e. the fragments that have received the higher

weight) are selected. Then all past queries Q, $Q \in \bigcup_i \mathcal{Q}_i$ receive a rank QR with respect to the top-n fragments:

$$QR(Q) = \frac{|F_Q \cap F_n|}{|F_Q|} * \frac{|F_Q \cap F_n|}{n}, \quad (5)$$

where F_Q represents the fragments of query Q. In other words, the queries are ranked based on a normalized metric measuring the number of common query fragments of each query Q to the top-n list. Finally, the top-m ranked queries are used as the recommendation set.

5. QUERY PREPROCESSING

In order to create the fragment-based query and session vectors, we needed to preprocess the queries included in the query logs and decompose them. This process consists of two steps, namely query generalization and query parsing. In the first step, the queries are generalized based on a set of rules, in order to be analyzed and matched more efficiently. Then, they are parsed and converted into a template, in preparation for comparison analysis.

5.1 Query Relaxation

Because of the plethora of slightly dissimilar queries existing in the query logs, we decided to relax them in order to increase their cardinality, and thus the probability of finding similarities between different user sessions. Our intuition is that if two users query the same table and attributes, using slightly different filtering conditions, the algorithm should consider them as similar.

As part of this relaxing process, we follow a simplified version of the framework proposed in [7]. In essence, all the WHERE clauses are relaxed by converting the numerical data and string literals to generic string representations. For example, all strings are replaced by STR, all hexadecimal numbers by HEXNUM and all decimals by NUM. A similar generalization is also followed for lists or ranges of numbers and strings. The mathematical and set comparators are also replaced by string equivalents, for example "=" is replaced by EQU and "≤" by COMPARE . In the current implementation of QueRIE we do not treat different numeric intervals as separate, however this is orthogonal to the framework and part of our future work plans.

5.2 Query Parsing

Once the queries are generalized, they are converted into fragments. The current implementation of QueRIE only supports SPJ (SELECT, PROJECT, JOIN) queries, whereas if a query includes sub-queries, these are dropped. However, this is an implementation detail orthogonal to the overall framework, which can be easily extended to support subqueries. Each of the SPJ fragments are separated using regular expressions. The Start and End designated keywords used to identify fragments are shown in Table 1.

Each distinct fragment is assigned a numerical identifier, used in the query and session vector representation. For each new fragment not previously recorded in the query log, QueRIE generates a new identifier. Such updates occur in real-time, as the current user posts a query including new fragments. In the case of the WHERE clause, only the joins and the filter conditions are stored. Because of the generalization, the fragments in the WHERE clause are not differentiated based on their actual values, rather based on the attributes used for filtering. For example, $s.x \geq 0.2$ and $s.x \geq 0.8$ will be represented by the same fragments. In addition we do not differentiate (i.e. handle differently) between joins and filters, as we anticipate the similarity calculation would generate proper results regardless of the type of WHERE condition.

Table 1: Parsing keywords

Fragment name	Start keyword	End keyword
Attribute string	SELECT	FROM
Relation string	FROM	WHERE, GROUP BY, ORDER BY, end of query
Where string	WHERE	GROUP BY, ORDER BY, end of query
Group By string	GROUP BY	ORDER BY, HAVING, end of query
Having string	HAVING	ORDER BY, end of query

Table 2: Data Set Statistics

# Sessions	180
# Distinct queries	1400
# Distinct query fragments	755
# Non-zero pair-wise fragment similarities	30436

6. EXPERIMENTAL EVALUATION

The framework proposed in this paper has been implemented in a prototype that will be demonstrated in [1]. In this section we present preliminary experimental results using real user traces. More specifically, we evaluate several parameters of the framework, namely the value of items used for the generation of recommendations n, the effect of the mixing factor α, and the employed weighted schemes.

6.1 Data Set.

We evaluated our framework using traces of the Sky Server database[3]. The traces contain queries posed to the database between the years 2006 and 2008. The query logs are anonymous, thus we used the methods described in [12] to clean and separate the query logs in sessions. For this reason, each session is considered as a different user. The characteristics of the data set and the queries are summarized in Table 2. Two real user sessions including a total of eight queries is included in [1].

6.2 Methodology.

In order to measure the prediction accuracy of QueRIE, we use the holdout set methodology [8]. The data is divided into two disjoint sets, the training set and the test set. The pair-wise fragment similarity is computed against the training set. Each user session in the test set is divided in two parts. One part is treated as the active user's queries, while the second part is treated as unseen (i.e. future) queries. Subsequently, using the active user's queries from the test set and the pre-calculated fragment-based similarities, QueRIE generates a set of query recommendations. We compare the recommended queries with the unseen queries from the test set and calculate the precision, recall and F-score for each session. This is performed by calculating these measures for each pair of queries, as shown in Equations 6, 7 and 8 and keeping the maximum value, assuming that the end user will also select only one out of the m recommended queries each time.

$$Precision = \frac{|F_r \cap F_u|}{|F_r|} \quad (6)$$

$$Recall = \frac{|F_r \cap F_u|}{|F_u|} \quad (7)$$

[3]We used the BestDR6 version.

Table 3: Default parameter values

Top-k	5
Top-n	5
Top-m	5
α	0.5
# weighting scheme	weighted (cosine)
Training set	160 sessions
Test set	20 sessions

$$F - Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (8)$$

In the formulas above, F_r and F_u represent the fragments of the recommended and unseen queries respectively. In the experiments that follow, we report the average precision and recall over the 160 sessions of the data set.

We performed several experiments evaluating the performance of the framework, and the effect of the various parameters of the algorithm. Due to space constraints, in this paper we present the most important findings in terms of the number of fragments n selected from S_0^{pred} to calculate the query rank QR, the mixing factor α, and the weighting scheme. Table 3 shows the default values kept constant for the remaining parameters in each case.

6.3 Experimental Results

Evaluation of the Top-n parameter. The recommended queries in QueRIE are identified, by first selecting the top-n fragments of the predicted summary S_0^{pred} and using them to rank all previous queries using the QR formula. Figures 1 and 2 show the average precision and F-score for various top-n values ($n \in \{3, 5, 10, 50\}$). We notice that the accuracy of the recommendations increases, as expected, with the value of n. However, for very large values of n, the accuracy decreases again. This is completely justifiable, since when n is a very large number, the notion of "most similar" fragments does no longer hold and barely similar items are included in the recommendation process. QueRIE achieves the higher precisions for $n = 10$ and $n = 5$ (0.8 and 0.75 respectively), whereas F-score is the same for both values (0.76 and 0.75 respectively). Given the small difference in terms of accuracy and the fact that the lower the number of fragments n, the faster the real-time calculations, we adopt $n = 5$ as the default value for the framework.

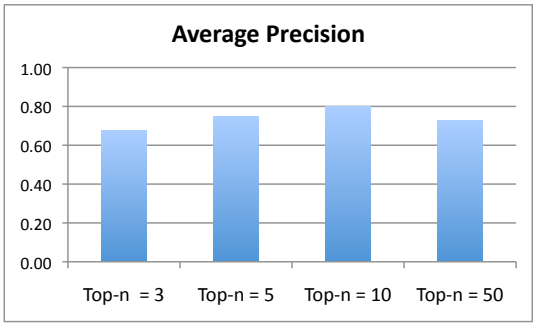


Figure 1: Average precision for various top-n values

Evaluation of the mixing factor α. In this instantiation of the QueRIE framework, the mixing factor α is introduced before the selection of the top-n fragments. Since including only the current user's session ($\alpha = 1$) is a content-based approach not employing the collective intelligence recorded in the query logs, we only evaluate the impact of including ($\alpha = 0.5$), or excluding ($\alpha = 0$) the fragments recorded in the active user's session. As expected intuitively, the accuracy of the recommendations is enhanced significantly when the active user's fragments are incorporated in the recommendation process. More specifically, precision and F-score are 0.76 and 0.74 respectively for $\alpha = 0.5$, whereas they drop to 0.48 and 0.43 when $\alpha = 0$, as shown in Figures 3 and 4. This verifies our initial claim that database recommender systems are very different in nature from their web counterparts. As pointed out in Section 3, one significant difference is that, in the case of SQL queries we want to expand or enhance the queries that were previously submitted by the user. The user benefits from this addition, since probably most users are interested in posting queries similar to the ones they have already posted during the same session.

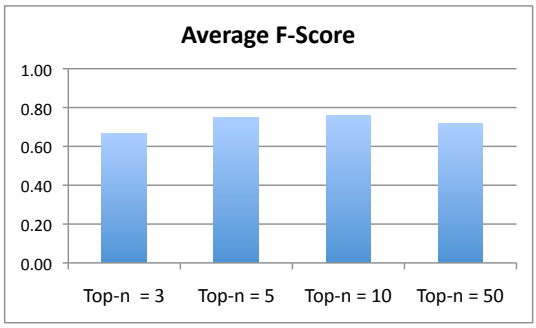


Figure 2: Average f-score for various top-n values

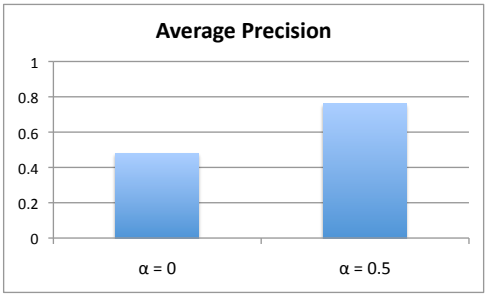


Figure 3: Average precision for different α values

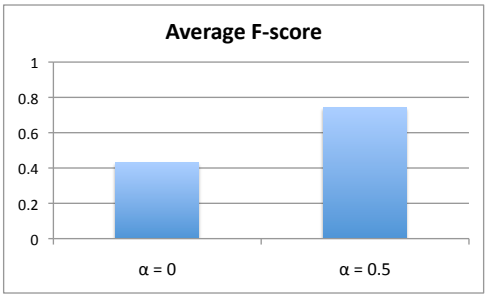


Figure 4: Average f-score for different α values

Evaluation of the weighting scheme. Depending on the weighting scheme selected, the representation of the query and session vectors, and consequently the metrics used to calculate the similarities between fragments, differs. In this instantiation we have

introduced the *binary* and the *weighted* schemes and employ the Jaccard coefficient and the cosine similarity metric respectively. In this set of experiments, we evaluate the effect of the representation. Intuitively, the binary representation is much more simplistic and is expected to provide less accurate results, since valuable information with regards to the importance of each fragment in a session is missing. The results, shown in Figures 5, 6 verify this intuition, however we notice that the difference is very small, with a precision of 0.74 and 0.79 for the binary and weighted schemes respectively, and an F-score of 0.69 and 0.74 respectively. For the specific dataset both schemes performed similarly in terms of real-time performance. Thus we adopt the *weighted* scheme as the default value, since it resulted in slightly better prediction accuracy.

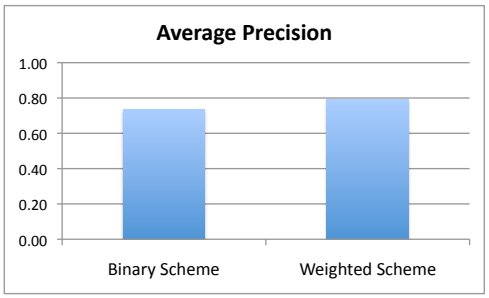


Figure 5: Average precision for different weighting schemes

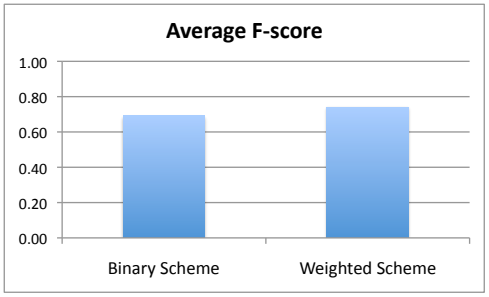


Figure 6: Average f-score for different weighting schemes

7. CONCLUSIONS

In this paper we presented a fragment-based instantiation of QueRIE, a recommender system that assists users when interacting with large database systems. QueRIE enables users to query a relational database, while generating real-time personalized query recommendations for them. We also performed an experimental evaluation of various parameters of the framework using real traces from the SkyServer database.

Overall, we showed that the precision of the recommendations is close to 80% when the active user's session is included in the prediction process, we employ the weighted scheme, and top-$n \in \{5, 10\}$ (with all other parameters set to default). This shows that QueRIE is very effective in generating useful recommendations to the end users of relational database systems. In terms of performance, QueRIE's fragment-based recommendation engine is able to generate real-time recommendations in quite fast (an average of 25 sec for each session in the test set).

We are currently working on evaluating the remaining parameters of the problem. We also plan to compare the fragment-based instantiation with the tuple-based one, proposed in our previous work.

8. ADDITIONAL AUTHORS

9. REFERENCES

[1] J. Akbarnejad, G. Chatzopoulou, M. Eirinaki, S. Koshy, S. Mittal, D. On, N. Polyzotis, and J. S. V. Varman. SQL QueRIE Recommendations (demo paper). In *Proc. of the 36th International Conference on Very Large Data Bases (VLDB 2010)*, 2010.

[2] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Collaborative filtering for interactive database exploration. In *Proc. of the 21st International Conference on Scientific and Statistical Database Management (SSDBM '09)*, 2009.

[3] A. Giacometti, P. Marcel, and E. Negre. Recommending Multidimensional Queries. In *Proc. of the 11th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'09)*, 2009.

[4] A. Giacometti, P. Marcel, E. Negre, and A. Soulet. Query recommendations for olap discovery driven analysis. *Intl. Journal on Data Warehousing and Mining (IJDWM) (to appear)*.

[5] A. Giacometti, P. Marcel, E. Negre, and A. Soulet. Query recommendations for olap discovery driven analysis. In *Proc. of the ACM 12th International Workshop on Data Warehousing and OLAP (DOLAP'09)*, 2009.

[6] N. Khoussainova, M. Balazinska, W. Gatterbauer, Y. Kwon, and D. Suciu. A case for a collaborative query management system. In *Proc. of the 4th Biennal Conference on Innovative Data Systems Research (CIDR 2009)*, 2009.

[7] N. Koudas, C. Li, A. K. H. Tung, and R. Vernica. Relaxing join and selection queries. In *Proc. of the 33nd international conference on Very large data bases (VLDB '06)*, pages 199–210, 2006.

[8] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data*. Springer, 2nd edition, 2007.

[9] S. Mittal, J. S. V. Varman, G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. QueRIE: A Recommender System supporting Interactive Database Exploration. In *2009 edition of the IEEE International Conference on Data Mining series (ICDM'09) - to appear in ICDM 2010 proceedings because of editor's error*, 2009.

[10] B. Mobasher. *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *LNCS*, chapter Data Mining for Personalization, pages 90–135. Springer, Berlin-Heidelberg, 2007.

[11] A. Simitsis, G. Koutrika, and Y. Ioannidis. Precis: From unstructured keywords as queries to structured databases as answers. *VLDB Journal*, 17(1):117–149, 2008.

[12] V. Singh, J. Gray, A. Thakar, A. S. Szalay, J. Raddick, B. Boroski, S. Lebedeva, and B. Yanny. Skyserver traffic report - the first five years. *Microsoft Research, Technical Report MSR TR-2006-190*, 2006.

[13] K. Stefanidis, M. Drosou, and E. Pitoura. "You May Also Like" Results in Relational Databases. In *3rd International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases (PersDB 2009)*, 2009.

[14] X. Yang, C. M. Procopiuc, and D. Srivastava. Recommending join queries via query log analysis. In *25th International Conference on Data Engineering (ICDE 2009)*, pages 964–975, 2009.

Re-ranking Web Service Search Results Under Diverse User Preferences

Dimitrios Skoutas
L3S Research Center
Hanover, Germany
skoutas@L3S.de

Mohammad Alrifai
L3S Research Center
Hanover, Germany
alrifai@L3S.de

Wolfgang Nejdl
L3S Research Center
Hanover, Germany
nejdl@L3S.de

ABSTRACT

Web service discovery aims at finding available services that match a given service description. This involves mainly the matchmaking of the functional parameters of the services, whereas non-functional attributes can also be considered and aggregated in the matching score of a candidate service as additional criteria for ranking the results. In this paper, we address the problem of re-ranking discovered services that include nominal attributes in their descriptions in order to satisfy users with diverse preferences. We present an approach to diversify the search results combining the degree of match on functional parameters with a method to achieve good coverage with respect to the values of nominal attributes. An evaluation on a publicly available dataset of Semantic Web services is also presented.

1. INTRODUCTION

Web service discovery aims at finding services whose description matches that of a desired service. The description of a service contains a functional and a non-functional part. The former provides information about what the service does and how it works. This is basically expressed in terms of the required inputs and generated outputs, as well as any pre-conditions that need to be satisfied in order for the service to be executed and any effects that result from its execution. Several methods exist for the matchmaking of functional service parameters. More traditional techniques include the application of string similarity measures on the parameter names, whereas, in the case of services in the Semantic Web, they mainly rely on logic-based match between concepts in an ontology that annotate service parameters; moreover, combinations thereof have also been proposed. The result is typically a score indicating the degree of match between the service request and the service advertisement, which is used to rank the discovered services.

The non-functional part of a service description may include information about the service provider and quality of service (QoS) parameters, such as price, response time, availability or reputation. These attributes can also be used during the service discovery and selection process as additional criteria to improve the ranking of the match results. A typical case is to use such information to resolve ties. For example, if two services have the same functional degree of match, then the one with the lowest cost or response time is preferred. Alternatively, an aggregate degree of match can be calculated, possibly using different weights on the various parameters. However, such solutions focus on numerical attributes, for which a total ordering exists, since these can be more easily handled and incorporated in the matchmaker. For example, it can be rather safely assumed that all users would prefer services that have cost and response time as low as possible or availability and reputation as high as possible.

However, some of these attributes contained in the service descriptions are nominal attributes, which can not be seamlessly incorporated in the matchmaker, since for them it is not possible to specify an ordering. On the contrary, different users, or even the same user in different contexts, may have different preferences regarding the values of these attributes. Typical examples include the provider of a service, the communication or security protocols a service may support, accepted file formats or different algorithms the service may employ to solve a specific problem. Matching such types of attributes is not straightforward, since the match depends on the user preferences for the values of these attributes. However, gathering explicit or implicit knowledge about the preferences of users on the Web is often very difficult or even impossible.

In this paper, we present a method for the discovery and selection of Web services focusing on attributes for which an ordering of the values is not defined, but instead it depends on the preferences of the user. The main idea underlying our approach is to increase the diversity in the search results in order, consequently, to increase the probability of satisfying users with different preferences. Diversifying search results has already been investigated in the context of document search on the Web [9, 6, 22]. Proposed solutions rely essentially on introducing more dissimilar documents in the result set, finding an appropriate balance between the relevance of the results and their dissimilarity. Our approach follows the same direction, but it proposes a different diversification objective that emphasizes on selecting representative results that provide good coverage of the whole list of matches. Moreover, Web service descriptions are complex objects; hence, simpler models, such as a bag-of-words model, which often work well for documents, are not appropriate for Web services search.

VLDB '10, September 13-17, 2010, Singapore

The main goal and advantage of this approach is to allow for personalized results in a flexible way and minimizing the burden to the user. One possibility for personalized search would be to request each user to create a profile with his/her preferences. However, this is not always desired; moreover, these preferences may change over time or may depend on a particular information need (e.g., a preference for a service provider). Alternatively, the system could ask the user to indicate his/her preferences at query time. However, this is cumbersome, and in addition, the user may not be aware in advance of the possible options. Instead, the described approach constitutes a two-step process to personalizing the results. First, starting from a potentially underspecified request, the system retrieves a diverse set of results aiming at covering the available options as much as possible. Then, once the user indicates a preferred result, a nearest neighbor query can be issued to retrieve more similar services. Hence, the user preferences are indicated implicitly, dynamically, and with minimum overhead.

In summary, our main contributions are listed below.

- We propose a method for re-ranking Web service search results to satisfy users with diverse preferences.
- We propose a diversification objective that favors representative services to achieve good coverage of the result set.
- We show how the diversification objective can be computed based on the different types of attributes in the service descriptions.
- We present an experimental evaluation of our approach on a collection of Semantic Web services.

The rest of the paper is structured as follows. Section 2 presents related work. Section 3 introduces our method for diversifying Web service search results. Section 4 presents our experimental evaluation. Section 5 concludes the paper.

2. RELATED WORK

Service discovery. Several approaches have been proposed for matching a Web service request with available service descriptions. These approaches can be categorized in two main families. The first -more traditional ones, based on WSDL and UDDI standards- follow IR-style search, performing a keyword-based search on the textual descriptions of the services or comparing the parameter names in the service descriptions using some standard string similarity measure [8]. The second family of approaches considers services in the Semantic Web, where service parameters are annotated using concepts from domain ontologies. Then, the matchmaking between service parameters is transformed to logic-based match between the corresponding ontology concepts, i.e., inferring, by means of an ontology reasoner, whether the denoted ontology classes are equivalent, superclass or subclass of each other, or disjoint [17, 16, 7, 20, 4]. To combine the advantages of both categories and to address their limitations, hybrid approaches have also been proposed and implemented. These compute aggregate scores based on both logic-based matching and string-based parameter similarities [14, 12]. Finally, ranking of services based on dominance relationships computed over multiple matching criteria has been proposed in [19].

In addition, there exist some approaches that involve the user in the service discovery process. The approach presented in [3] employs ontologies and user profiles, and uses techniques such as query expansion or relaxation to better satisfy user requests. The work in [23] focuses on QoS-based Web service discovery, proposing a reputation-enhanced model. Reputation scores are assigned to the services by a reputation manager based on user feedback regarding their performance. Then, a discovery agent uses the reputation scores for service matching, ranking and selection. User preferences, expressed in the form of soft constraints, are applied for Web service selection in [13], focusing on the optimization of preference queries. Utility functions are used in [15] to model service configurations and associated user preferences for optimal service selection. In [8], different types of similarity for service parameters are combined using a linear function, with weights being assigned manually. It is mentioned that the weights can be learned from user feedback, but this is not addressed.

Our approach defers from the above works mainly in two aspects. First, the aforementioned approaches deal with numerical QoS parameters, for which a global and total ordering exists; hence, they can more easily be incorporated to and combined with other criteria for service selection and ranking. Instead, we focus in this paper on QoS parameters with non-numeric values (e.g., text or categorical data), for which no ordering can be defined. Second, the approaches above assume that a user profile, preferences or feedback is available for the service requestor. However, collecting such implicit or explicit information for Web users at large is not practical, if not infeasible. Nevertheless, the fact that different users have different preferences and needs still needs to be taken into account. Our approach addresses this issue by re-ranking the match results to increase their diversity, and, hence, to reduce the risk that for a given user there is not *any* result that meets his/her preferences.

Diversification of Web search results. Recently, the problem of diversifying Web search results has received a lot of attentions as a means to satisfy users on the Web with diverse preferences and information needs. The problem is typically formulated as optimizing an objective function that specifies a trade-off between the relevance of the returned documents with respect to the given query and the dissimilarity among these documents [9]. Essentially, this is similar to the idea of the Maximal Marginal Relevance criterion, which has been proposed in [6] for re-ranking the query results and it is also applied often for document summarization. It combines query relevance and novelty of information, by measuring the dissimilarity of a search result with respect to the ones before it in the ranked result list. In a similar direction, [22] addresses the problem of re-ranking search results adopting the idea of Modern Portfolio Theory from the field of finance. It considers documents not individually but in combination with other documents, formulating the problem as a portfolio selection problem. Results are then selected to maximize the relevance, while minimizing the variance, where the notion of variance corresponds, inversely, to that of diversity. The problem of diversifying query results from a database has been considered in [21]. However, that approach assumes that there is a known ordering of the user preferences with respect to the involved attributes. For example, for a query for "cars", the returned tuples should be first diversified with respect

to the car model, then to price and then to color. Moreover, we propose and apply a diversification objective that targets the selection of more representative results in order to achieve better coverage.

The main difference of our approach is that it deals with Web service search results rather than documents. Service descriptions are complex objects, comprising parameters of different types (functional and non-functional, numeric and non-numeric) that need to be handled accordingly.

3. DIVERSIFYING DISCOVERED SERVICES

3.1 Diversification Objective

Assume a service request R and a repository containing a set of service advertisements $\mathcal{S}$. To discover services that match the request R, a matchmaker is invoked, which applies one or more matching criteria to calculate a *degree of match* (*dom*) between R and each available service $S \in \mathcal{S}$ (see Section 2 for more details). The core of this operation is to match the input and output parameters in the two descriptions, although pre-conditions and effects or other parameters can also be taken into account. Then, the found matches are sorted according to their degree of match and the ranked list (or the top-k matches) is returned to the user. However, this list may often contain services that are very similar to each other, hence being biased toward the needs and preferences of only a subset of users. For this reason, the goal is to re-rank the list of results in order to include services that are still relevant to the request but less similar to each other.

Let $sim : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ be a function that computes the similarity between two service descriptions. Notice that the two functions, *dom* and *sim*, serve very different purposes. Function *dom* checks, for example, whether the outputs of the advertised service "fulfill" the outputs of the request, whereas function *sim* checks, for example, whether two services have the same provider or whether they support the same protocols or file formats. Then, the goal is to re-rank the discovered services so that the top-k results have a degree of match to the query that is as high as possible, while at the same time being as dissimilar to each other as possible. However, these two objectives are often contradictory, since a new service that is introduced to make the result list more diverse may have lower degree of match than the one it replaces. Therefore, these two factors need to be balanced using an objective function.

A diversification objective, referred to as MaxMin, has been proposed in [9] for diversifying search results of keyword queries on the Web. Applying this objective would return a set of k services that maximize the minimum degree of match and the minimum dissimilarity in the set. Formally, it selects the set of top-k results that maximizes the function:

$$f(\mathcal{S}_k) = \lambda \cdot \min_{S \in \mathcal{S}_k} dom(R, S) + \min_{S_1, S_2 \in \mathcal{S}_k} dist(S_1, S_2) \quad (1)$$

where distance (i.e., dissimilarity) is measured by $dist(S_1, S_2) = 1 - sim(S_1, S_2)$ and $\lambda > 0$ is a parameter that specifies the relative emphasis between the two factors.

The drawback of this diversification objective is that it measures the degree of match and the dissimilarity only *within* each candidate top-k list of results, ignoring the remaining ones, i.e., the ones below rank k. This has the side effect that it is biased towards more extreme rather than more representative cases. In other words, it might give priority to outliers. Assume, for example, a list of matches containing two services that have a high degree of match to the request and are very dissimilar to each other with respect to other characteristics. Then, this objective would favor these services for the top 2 results. However, these are not necessarily representative of the rest of the matches.

To address this issue, we propose a different diversification objective that aims also at providing good *coverage* of the whole result list, selecting services that are good representatives of the whole result set (and, consequently, also diverse with respect to each other). The main idea is that, in order to achieve good coverage, for each identified match there should be at least one service in the top-k list that is sufficiently similar to it. Moreover, since this apparently can not be achieved for all the services in the result list, priority should be given to those services with higher degree of match to the request. That is, if a service has a very high degree of match to the query, it should be well represented in the top-k results, either by including itself or by including one that is very similar to it. Notice however that this does not mean necessarily a higher representation, in terms of number of services, in the output set for the more relevant services. Indeed, if the services with high degrees of match happen to be very similar to each other, then they can be adequately covered with just a few representatives, or even a single one. Hence, this method is robust with respect to the problem of diminishing returns, as pointed out in [1], which leads to decreased user satisfaction.

We now formalize this diversification objective. Given a subset $\mathcal{S}_k$ of the query results, and a service S, we define the *coverage error* of S with respect to $\mathcal{S}_k$ as the minimum distance between S and any of the services in $\mathcal{S}_k$, i.e.:

$$cerr(S, \mathcal{S}_k) = \min_{S' \in \mathcal{S}_k} dist(S, S') \quad (2)$$

Lower values mean that there exists a selected service that is highly similar to the considered one. Hence, the quality of $\mathcal{S}_k$ is characterized by its maximum error in representing the matched services for the given request. Moreover, the coverage error for a service should be weighted according to its degree of match. If a service has a high degree of match to the request but is poorly covered, i.e., it is neither included in the top-k results nor there exists another sufficiently similar service in the top-k list, this should incur a higher penalty. Hence, this objective selects the subset of query results that *minimizes* the following function:

$$f(\mathcal{S}_k) = \max_{S \in \mathcal{S}} \{dom(R, S)^\lambda \times cerr(S, \mathcal{S}_k)\} \quad (3)$$

where the parameter λ determines, as previously, the trade-off between the two factors, namely the degree of match and the diversity of the results.

We refer to this diversification objective as MaxCov. Notice that the score of the set $\mathcal{S}_k$ in Equation 3 depends on *all* the matched services for the query (max is computed over all the services in $\mathcal{S}$), while in Equation 1 it depends only on the top-k subset (min is computed over $\mathcal{S}_k$).

Example. We present a simple example to illustrate the difference between the MaxMin and the MaxCov objectives. Assume a service request R and a set of 6 relevant services S_1, S_2, ..., S_6, with distances from each other as

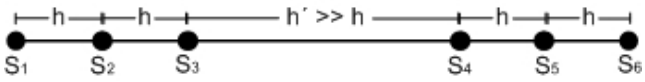


Figure 1: Illustrative example showing the difference between the MaxMin and MaxCov objectives

depicted in Figure 1. Assume also that we want to select as answer to this request a subset comprising $k = 2$ of these services. Disregarding the factor of the degree of match, i.e., assuming that all these services equally match the request, the MaxMin objective, which is driven by the pair-wise distances of the objects within the result set, selects as output the set $M_1 = \{S_1, S_6\}$. These are indeed the most dissimilar services. On the other hand, the MaxCov objective, which considers the distances of the selected objects to the whole result set, selects as output the subset $M_2 = \{S_2, S_5\}$. These can indeed be considered as better representatives.

3.2 Computing DoM and Similarity

In the following we discuss how to compute the degree of match *dom* between a service request R and a set of available service descriptions $\mathcal{S}$ and the similarity between two service descriptions S_1 and S_2.

As presented in Section 2, there exists several methods for matching a service request R with a service advertisement S. This operation is based on matching functional service parameters, typically inputs and outputs. Recently a lot of efforts have focused on the discovery of services in the Semantic Web and three main languages have been proposed to semantically mark up service descriptions, namely WSDL-S [2], OWL-S [5] and WSMO [11]. The main idea underlying all these approaches is to associate service parameters with classes in a domain ontology $\mathcal{O}$. This allows both the service provider and the user searching for a service to describe the intended meaning of the parameters in an unambiguous way, which facilitates the automation of the discovery process and increases the precision w.r.t. plain keyword search, where ambiguity, synonyms and homonyms need to be taken into account. In this setting, a reasoner is employed to infer the relationship between the corresponding ontology classes (i.e., equivalence, subsumption, disjointness) and based on that the type of match is determined accordingly (e.g., exact, plug-in, subsumes, subsumed-by or fail) [17].

For simplicity, we consider here only input and output parameters. A service advertisement S matches a service request R if (a) the outputs requested by R are matched by those offered by S and (b) the inputs required by S are provided in R. To quantify the degree of match, a method that considers the "proximity" of classes in the ontology can be applied [20, 18]. More specifically, the degree of match between two parameters can be measured by means of the common super classes of the corresponding classes C_1 and C_2 in the ontology $\mathcal{O}$, as follows:

$$dom(C_1, C_2) = \frac{|\{C \mid C \sqsubseteq C_1 \wedge C \sqsubseteq C_2\}|}{\texttt{max}(|\{C \mid C \sqsubseteq C_1\}|, |\{C \mid C \sqsubseteq C_2\}|)} \quad (4)$$

Then, the degree of match between the request and the advertisement is computed by aggregating the degrees of match of individual parameters. This can be extended to other parameters, apart from inputs and outputs.

As a metric for computing the similarity between two service descriptions S_1 and S_2 we use the Jaccard similarity, which is also commonly used in information retrieval for measuring the similarity between two documents.

The Jaccard similarity (also known as the Jaccard Coefficient) between two sample sets A and B is defined as the size of the intersection divided by the size of the union of the two sample sets, i.e.:

$$sim(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (5)$$

For computing the similarity between two services, we compare the non-functional part of the descriptions, and in particular nominal attributes, e.g. the set of supported protocols for transport, security, transactions etc.

Algorithm 1 Approximate algorithm for MaxCov

Input: : A service request R, the available service descriptions $\mathcal{S}$, an integer k
Output: : The re-ranked list $\mathcal{S}_k$ of top-k matches

```
1: d = arg max_{S∈S} dom(R, S)
2: S_k = {d}
3: for i = 1 to k − 1 do
4:     S = arg max_{S∈S} {dom(R, S)^λ × cerr(S, S_k)}
5:     S_k = S_k ∪ {S}
6: end for
7: return S_k
```

3.3 Diversification Algorithm

The MaxMin diversification objective presented in Section 3.1 corresponds to the vertex weighted version of the Minimum k-Center problem, which is known to be NP-hard [10] (for metric distances). Notice that the same also holds for the MaxMin objective specified in Equation 1 which was proposed in [9]. Consequently, one needs to resort to greedy approximation algorithms. For the MaxCov objective, a 2-approximation greedy algorithm can be derived from the Minimum k-Center problem. According to that, the re-ranking of services is done as specified in Algorithm 1. The algorithm works as follows. First, the service with the highest degree of match is selected. Then, the algorithm proceeds in $k-1$ iterations, selecting in each iteration the service with the maximum weighted coverage error with respect to those selected so far.

4. EXPERIMENTAL EVALUATION

In this section we describe our experimental evaluation of the proposed method for diversifying the results of Web service search.

For our experiments, we have used the OWLS-TC v2 collection[1]. This is a publicly available collection of Semantic Web services described in OWL-S and it is often used to evaluate and compare different matchmaking algorithms. It comprises 1007 services retrieved from public UDDI repositories. These services have then been described in OWL-S, using ontologies from 7 different domains to semantically annotate service input and output parameters. This collection provides also a set of 28 service requests and their corresponding relevance sets identified manually, in order to allow different matchmakers to compare the results based on the standard recall and precision measures. However,

[1] http://projects.semwebcentral.org/projects/owls-tc/

Table 1: Non-Functional Attributes

Attribute	Possible Values
Message Encoding	XML
	SOAP 1.1
	SOAP 1.2
	WS-Addressing
Security Protocol	WS-Security SOAPMessageSecurity 1.0
	WS-Security SOAPMessageSecurity 1.1
	WS-Security UsernameTokenProfile 1.0
	WS-Security UsernameTokenProfile 1.1
	WS-Security X.509 Certificate 1.0
	WS-Security X.509 Certificate 1.1
	WS-Security X.509 Kerperos 1.0
	WS-Security X.509 Kerperos 1.1
	WS-SecureConversationLanguage
	WS-TrustLanguage
	WS-ReliableMesseging 1.0
	WS-ReliableMesseging 1.1
Transport Binding Protocol	HTTP 1.1
	SOAP 1.1 HTTP Binding
	SOAP 1.2 HTTP Binding
Transaction Protocol	WS-Coordination 1.0
	WS-Coordination 1.1
	WS-Coordination 1.2
	WS-AtomicTransaction 1.0
	WS-AtomicTransaction 1.1
	WS-AtomicTransaction 1.2
	WS-BusinessActivity 1.0
	WS-BusinessActivity 1.1
	WS-BusinessActivity 1.2

this benchmark is not appropriate for our purpose, since it only indicates services that are relevant to the request w.r.t. input and output parameters without any consideration on the diversity of the results. Therefore, we have used the provided queries for our experiments but we do not conduct any evaluation in terms of recall and precision w.r.t. the provided relevant sets; instead, we want to measure the improvement in terms of the coverage error, as shown below. Moreover, the service descriptions included in this collection contain information only about input and output parameters, since these are the typical criteria taken into consideration by existing matchmakers, as described in Section 2. To overcome this limitation, we have extended the description of each Web service in the dataset with a vector of 4 non-functional attributes: `Message Encoding Schema`, `Security Protocol`, `Transport Binding Protocol` and `Transaction Protocol`. For each one of these attributes, we have identified a set of possible values, as shown in Table 1. Then, we have randomly assigned to each Web service in the collection a value for each of these attributes. For the implementation, we have used the OWL-S API[2] for parsing the OWL-S descriptions of the services in the collection.

In this evaluation, we wanted to measure how diverse are the results obtained by our *MaxCov* diversification method as described in Algorithm 1. As a measure for diversification we computed the objective value based on the coverage error (as defined in Equation 3) of the results set. Notice that lower values indicate lower coverage error and higher diversity in the results. For this purpose, we first computed the degree of match *dom* between each service in the collection and each query. The set of candidate matches for a certain query R consists of all the services that have a non-zero degree of match with R. In this experiment, we considered only queries that have at least 100 candidate matches, since considering coverage and diversity is less important when dealing with queries that have a relatively small result set. We then applied the following two methods for selecting the top-k results of each query:

1. `Top-K`: this is the "default" ranking, i.e., considering only the degree of match without taking into account any non-functional attributes and without applying any diversification method. The set of candidate matches are sorted by the *dom* value in descending order and the top-k services on the list are returned.

2. `MaxCov`: these are the top-k results returned by the method that applies Algorithm 1 for minimizing the objective function given in 3.

The degree of match between the query and the services was computed based on the input and output parameters as described in Section 3.2, while the similarity between the services was computed on the 4 aforementioned attributes, using Jaccard similarity, also described in Section 3.2.

[2]http://www.mindswap.org/2004/owl-s/api/

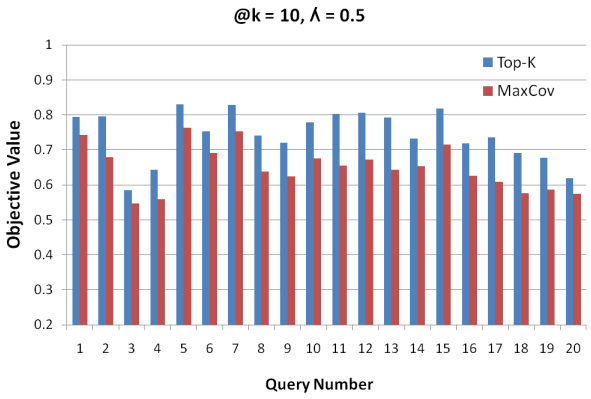


Figure 2: Comparison of Diversity Degree @ k = 10

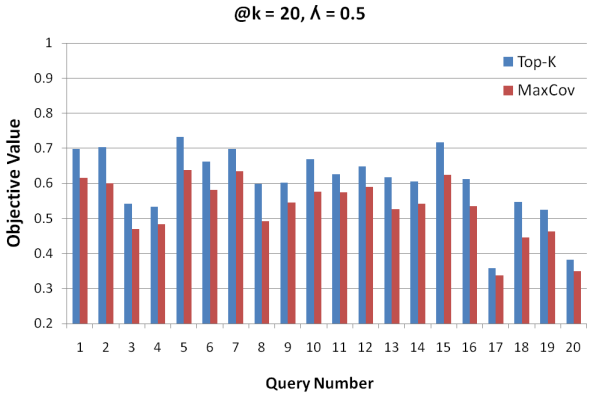


Figure 3: Comparison of Diversity Degree @ k = 20

In Figure 2 and Figure 3 we compare the objective value defined by Equation 3 of the selected top-k services for each query with $k = 10$ and $k = 20$, respectively. As discussed earlier in Section 3.1, the parameter λ is used to determine the trade-off between the degree of match and the diversity of the results. Note that in the objective function defined in Equation 3, λ is used as an exponent of the *dom* value and that the *dom* value is between 0 and 1. Therefore, the lower the value of λ, the higher the weight of the *dom* value in the objective function. In our experiments, we set the value of λ to 0.5, which gives the *dom* value more weight than the diversity. The results in Figure 2 and Figure 3 show that the objective value of the `MaxCov` method is lower than the objective value of the `Top-K` method for all queries. This indicates that the results obtained by the `MaxCov` method provide better coverage than those obtained by the other methods w.r.t. the whole set of candidate matches. We also observe that with $k = 20$ the average objective value of both methods is lower than with $k = 10$. The reason for this behavior is that the increase in the number of selected services increases the probability that more relevant services are represented by the top-k services, and hence lowering the coverage error, which in turn leads to a lower value of the objective function.

5. CONCLUSIONS

We have proposed a method to diversify Web service search results in order to deal with users on the Web that have different, but unknown, preferences. Our method focuses specifically on nominal attributes in service descriptions, for which a total ordering can not be defined, since it is dependent on the preferences of each particular user. Such attributes can not be easily incorporated in the matchmaking process, when computing a degree of match to the query. Instead, our approach relies on including diverse and representative services in the results to satisfy different users. We have presented a diversification method and we have evaluated the results on a collection of Semantic Web services.

Directions for future work include mainly the evaluation of our method using larger collections of services and, especially, service descriptions with a larger number of attributes. We would also like to conduct a user study to examine how user satisfaction increases when providing more diversity in the discovered services.

Acknowledgments

This work was partially supported by the FP7 EU Projects LivingKnowledge (contract no. 231126) and Sync3 (contract no. 231854).

6. REFERENCES

[1] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM*, pages 5–14, 2009.

[2] R. Akkiraju and et. al. Web Service Semantics - WSDL-S. In *W3C Member Submission*, Nov. 2005.

[3] W.-T. Balke and M. Wagner. Cooperative Discovery for User-Centered Web Service Provisioning. In *ICWS*, pages 191–197, 2003.

[4] U. Bellur and R. Kulkarni. Improved Matchmaking Algorithm for Semantic Web Services Based on Bipartite Graph Matching. In *ICWS*, pages 86–93, 2007.

[5] M. Burstein and et. al. OWL-S: Semantic Markup for Web Services. In *W3C Member Submission*, Nov. 2004.

[6] J. G. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998.

[7] J. Cardoso. Discovering Semantic Web Services with and without a Common Ontology Commitment. In *IEEE SCW*, pages 183–190, 2006.

[8] X. Dong, A. Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Similarity Search for Web Services. In *VLDB*, pages 372–383, 2004.

[9] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, pages 381–390, 2009.

[10] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

[11] H. Lausen, A. Polleres, and D. Roman (eds.). Web Service Modeling Ontology (WSMO). In *W3C Member Submission*, June 2005.

[12] F. Kaufer and M. Klusch. WSMO-MX: A Logic Programming Based Hybrid Service Matchmaker. In *ECOWS*, pages 161–170, 2006.

[13] W. Kießling and B. Hafenrichter. Optimizing Preference Queries for Personalized Web Services. In *Communications, Internet, and Information Technology*, pages 461–466, 2002.

[14] M. Klusch, B. Fries, and K. P. Sycara. Automated Semantic Web service discovery with OWLS-MX. In *AAMAS*, pages 915–922, 2006.

[15] S. Lamparter, A. Ankolekar, R. Studer, and S. Grimm. Preference-based selection of highly configurable web services. In *WWW*, pages 1013–1022, 2007.

[16] L. Li and I. Horrocks. A Software Framework for Matchmaking based on Semantic Web Technology. In *WWW*, pages 331–339, 2003.

[17] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic Matching of Web Services Capabilities. In *ISWC*, pages 333–347, 2002.

[18] D. Skoutas, D. Sacharidis, V. Kantere, and T. K. Sellis. Efficient semantic web service discovery in centralized and p2p environments. In *ISWC*, pages 583–598, 2008.

[19] D. Skoutas, D. Sacharidis, A. Simitsis, and T. Sellis. Ranking and clustering web services using multi-criteria dominance relationships. In *IEEE Trans. on Services Computing (to appear)*, 2010.

[20] D. Skoutas, A. Simitsis, and T. K. Sellis. A Ranking Mechanism for Semantic Web Service Discovery. In *IEEE SCW*, pages 41–48, 2007.

[21] E. Vee, U. Srivastava, J. Shanmugasundaram, P. Bhat, and S. Amer-Yahia. Efficient computation of diverse query results. In *ICDE*, pages 228–236, 2008.

[22] J. Wang and J. Zhu. Portfolio theory of information retrieval. In *SIGIR*, pages 115–122, 2009.

[23] Z. Xu, P. Martin, W. Powley, and F. Zulkernine. Reputation-Enhanced QoS-based Web Services Discovery. In *ICWS*, pages 249–256, 2007.

Adapting Generic Web Structures with Semantic Web Technologies: A Cognitive Approach

Mario Belk
University of Cyprus
75 Kallipoleos, 1678 Nicosia, Cyprus
+35722892700
belk@cs.ucy.ac.cy

Panagiotis Germanakos
University of Cyprus & Nicosia
75 Kallipoleos, 1678 Nicosia, Cyprus
+35722892700
pgerman@cs.ucy.ac.cy

Nikos Tsianos
University of Athens
5 Stadiou St., GR-10562, Athens, Greece
+302103689282
ntsianos@media.uoa.gr

Zacharias Lekkas
University of Athens
5 Stadiou St., GR-10562, Athens, Greece
+302103689282
zlekkas@gmail.com

Costas Mourlas
University of Athens
5 Stadiou St., GR-10562, Athens, Greece
+302103689282
mourlas@media.uoa.gr

George Samaras
University of Cyprus
75 Kallipoleos, 1678 Nicosia, Cyprus
+35722892700
cssamara@cs.ucy.ac.cy

ABSTRACT

The research described in this paper focuses on adapting generic Hypermedia Environments with the use of Semantic Web technologies. The use of machine understandable semantics in Web applications is increasing significantly, ranging from semantic search queries to personalized product recommendations. Given the significance of human factors in Web structures' transformations, we propose an Ontological Web Personalization and Adaptation Mechanism based on users' cognitive parameters. This mechanism consists of a number of interrelated components; modeling users based on their cognitive parameters and providing content adaptation based on semantics. An RDFa schema has been designed that enables standard annotations in any XHTML Web-page, thus making structured data available for the adaptation process, but also for any service or tool that supports the same standard. Our work has been positively evaluated using an existing commercial Web-site that was filtered through the adaptation mechanism based on users' cognitive parameters.

1. INTRODUCTION

Advances in Web-based oriented technologies and services are taking place with a considerable speed around the world. As communications and IT usage become an integral part of many people's lives and the available products and services become more varied and sophisticated, users expect to be able to personalize a service to meet their individual needs and preferences.

Furthermore, the plethora of information and services as well as the complicated nature of most Web structures intensify the orientation difficulties, as users often lose sight of their original goal, look for stimulating rather than informative material, or even use the navigational features unwisely. As the eServices sector is rapidly evolving, the need for such Web structures that satisfy the heterogeneous needs of its users is becoming more and more evident [1].

In recent years, there has been a rapid growth in research and experiments that work on personalizing computer-mediated platforms, according to user needs and indeed, the challenges ranging in this area are not few. A challenge is to design a comprehensive and expressive user model composed of cognitive parameters that can be used in Web systems. Based on that user model, engineers will design and develop personalized and adaptive interfaces and software. This will enable easy access to any content while being sufficiently flexible to handle changes in users' context, perception and available resources, optimizing the content delivery while increasing their comprehension capabilities and satisfaction.

At a more technical level, a challenge is to study and design structure of meta-data (semantics) coming from the providers' side, aiming to construct a Web-based personalization mechanism that will serve as an automatic filter adapting the received hypertext/hypermedia content based on the comprehensive user profile. The final system will provide a complete adaptation and personalization Web-based solution to users satisfying their individual needs and preferences.

Henceforth, this paper describes a Personalization and Adaptation mechanism that consists of three main sections; i) description of an Ontological Cognitive User Model (OCUM) for modeling users' unique cognitive parameters that could be used in any online computer mediated application, returning an optimized adaptive result to users, ii) an adaptation mechanism that maps the OCUM with semantically annotated web objects based on an RDFa content schema for smart web objects, and iii) description of an RDFa content schema that enables standard annotations in any XHTML Web-page, thus making structured data available for the personalization and adaptation process, but also for any service or tool that supports the same standard. An evaluation of the expected impact of the content reconstruction concludes the paper.

2. BACKGROUND

Effective personalization of content involves two important challenges: accurately identifying users' comprehensive profiles and mapping any hypermedia content in such a way that enables efficient and effective navigation and presentation during the adaptation process.

Today's most popular Web-sites (http://www.alexa.com) like Google, Microsoft Live, Yahoo, Amazon, eBay, BBC news etc. primarily use customization techniques where users have direct control and explicitly select between certain options. In the same line, personalization is driven by the system which tries to serve up individualized pages to users according their profiles and

needs. Although, personalization is used by many of these popular Web-sites (especially Google), the techniques they maintain are lying under the predetermined customization of services or products and not to the actual personalization and dynamic reconstruction of content based on user preferences. User preferences might be extended beyond the traditional characterizations of users that might include intrinsic cognitive values that could be considered as the control factors for an efficient adaptation process.

Many search engines/systems, such as Google Personalized Search (http://www.goggle.com/psearch), build a user profile by means of implicit feedback where they adapt the results according to the search history of users. Many systems employ search personalization on the client-side by re-ranking documents that are suggested by an external search engine [2, 3]. Since the analysis of pages in the result list is a time consuming process, these systems often take into account only the top ranked results. Also, only the snippets associated with each page in the search results is considered as opposed to the entire page content.

One increasingly popular method to mediate information access is through the use of ontologies [4]. Researchers have attempted to utilize ontologies for improving navigation effectiveness as well as personalized Web search and browsing, specifically when combined with the notion of automatically generating semantically enriched ontology-based user profiles [5].

One such system is OntoSeek [6], which is designed for content-based information retrieval from online yellow pages and product catalogs. OntoSeek uses simple conceptual graphs to represent queries and resource descriptions. The system uses the Sensus ontology [7], which comprises a simple taxonomic structure of about 50,000 nodes.

Another similar system developed by Labrou and Finin [8] uses Yahoo! [9] as an ontology. The system semantically annotates Web pages via the use of Yahoo! categories as descriptors of their content. The system uses Telltale [10, 11, 12] as its classifier. Telltale computes the similarity between documents using n-grams as index terms. The ontologies used in the above examples use simple structured links between concepts.

A richer and more powerful representation is provided by SHOE [13, 14]. SHOE is a set of Simple HTML Ontology Extensions that allow WWW authors to annotate their pages with semantic expressed in terms of ontologies. SHOE provides the ability to define ontologies, create new ontologies which extend existing ontologies, and classify entities under an "is a" classification scheme.

Google has also recently announced (http://www.google.com/webmasters/tools) that their search engine is going to support enhanced searching in Web-pages, by using RDFa and Microformats embedded in XHTML. Google states that the extra (structured) data will be used in order to get results for Product Reviews (e.g. CNET Reviews), Products (e.g. Amazon product pages), People (e.g. LinkedIn profiles) and any other types of resources will be made public through the data-vocabulary.org (http://rdf.data-vocabulary.org/rdf.xml).

However, the abovementioned ontologies do not utilize user centric personalization approaches in the sense of considering intrinsic user values, such as cognitive characteristics for the adaptation of Web content.

3. A PROPOSED ONTOLOGICAL ADAPTATION MECHANISM

Web Personalization and Semantics are two research areas that attempt to provide solutions to user problems related to content navigation and presentation. They both employ specialized approaches and techniques for alleviating difficulties and constraints imposed by the Web.

Key factors that are employed by personalization mechanisms for filtering user profiles and adapting content accordingly are based on cognitive parameters. On the other hand, Semantics contribute to the whole adaptation process with machine understandable representation of user profiles and web content.

Therefore, the main scope of this section is to bring together the abovementioned considerations and propose an Ontological Adaptation Mechanism (OAM) (Fig. 1) that is composed of three main layers; i) User Profile Layer, ii) Adaptation Mapping Layer, and iii) Web Content Layer.

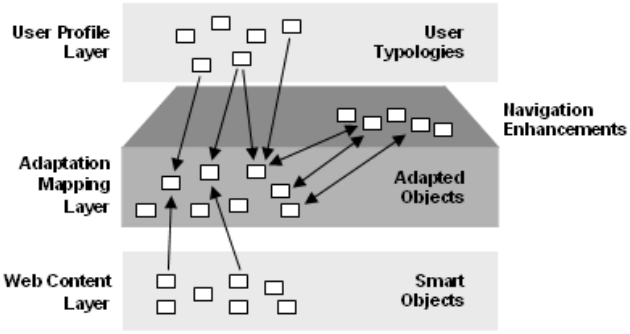


Figure 1. Ontological Adaptation Mechanism

The first layer of the OAM is the User Profile Layer; responsible to model users' cognitive typologies. In a high level view, each user typology is a semantically defined object (RDFa object) that contains all the intrinsic cognitive parameters of users. On the other end of OAM; the Web Content Layer model's any hypermedia Web content with specific meta-characteristics using again an RDFa vocabulary to annotate specific areas of an XHTML document as Smart Objects. The middle layer; the Adaptation Mapping Layer is responsible for mapping the User Typologies of the User Profile Layer with the Smart Objects of the Web Content Layer. Based on this mapping, all Smart Objects of an XHTML document are adapted (i.e. show content in a diagrammatical form in case of an Imager) and extra navigation enhancements are provided to the end-user.

For example, a user has been identified of a particular typology in the User Profile Layer. Furthermore, any semantically annotated Web content will be derived in the Web Content Layer. Finally, the Adaptation Mapping Layer will map the semantic user typology with the semantic Web content providing an adapted result as well as additional navigation enhancements.

The following sections will explain in more detail how each layer utilizes the theoretical conceptualization of the Ontological Adaptation Mechanism.

3.1 User Profile Layer

The User Profile Layer implements the Ontological Cognitive User Model (OCUM). OCUM is based on the theoretical conceptualization of a comprehensive model in the field of Web personalization and adaptation, which integrates cognitive parameters and attempts to apply them on a Web-based environment. The particular cognitive concepts have already been proposed by the authors and positively evaluated in the information space [15].

This model consists of an optimized series of cognitive parameters and tends to further enhance user profiles (considered the main filtering elements for Web personalization systems), that could be used in any hypertext computer-mediated platform in order to return a more enhanced user-centric result by reconstructing (adapting) any content coming from the provider.

The main uses of this model are: 1) to enable consistent implementation (and interoperation) of all hypertext computer-mediated systems that use human factors as their main filtering element, based on a shared background vocabulary, and 2) to play the role of a domain ontology that encompasses the core human factors elements for hypertext computer-mediated systems and that can be extended by any other individual or group.

Table 1 shows a sample of the OCUM's vocabulary. The full RDFa vocabulary can be found online in (http://www4.cs.ucy.ac.cy/adaptiveweb/rdf.xml).

Table 1. OCUM RDFa Vocabulary

```
<rdfs:Class rdf:ID="Person">
  <rdfs:comment>Represents a Person,
living/fictional.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#Resource"/>
</rdfs:Class>
<rdf:Property rdf:ID="name">
  <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>
<rdf:Property rdf:ID="title">
  <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>
<rdf:Property rdf:ID="affiliation">
  <rdfs:domain rdf:resource="#Person"/>
</rdf:Property>
….
<rdf:Property rdf:ID="imagerverbal">
  <rdfs:comment>The level of cognition of a person regarding
imager and verbal can be specified by a string literal or a Person
instance.</rdfs:comment>
  <rdfs:domain rdf:resource="#OCUM"/>
  <rdfs:range>
    <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
                <owl:Class rdf:about="#Person"/>
                <owl:Class rdf:about="xsd:string"/>
      </owl:unionOf>
    </owl:Class>
  </rdfs:range>
</rdf:Property>
```

This vocabulary consists of a number of classes and properties which describe a user's profile. With regards to the sample vocabulary shown in Table 1, the main class of this vocabulary is Person that represents a living or fictional person. The Person class has the following basic properties: i) "name" property; the Person's name, ii) "title" property; the Person's title (i.e. Prof. or Managing Director), iii) "affiliation" property; the Person's affiliation. A Person class has also the following properties with regards its Cognitive Style parameters: i) "imagerverbal" property; the level of cognition of the Person regarding imager and verbal, ii) "wholistanalyst" property; the level of cognition of the Person regarding wholistic or analyst, and iii) "workingmemory" property; the Person's working memory capacity.

In this respect, the Person class, for example, in the RDFa instance (Table 2) is the main entity. Specializations of the Person entity are the Cognitive Styles, the Working Memory and Personal Detail entities.

Table 2. RDFa Instance of a User's OCUM

```
<div xmlns:v="http://www4.cs.ucy.ac.cy/adaptiveWeb/rdf/#"
typeof="v:Person">
  <div>
      <span property="v:name">John Smith</span>
      <span property="v:title">Managing Director.</span>
      <span property="v:affiliation">AWeb Solutions</span>
  </div>
  <div>Cognitive Style
      <span rel="v:imagerverbal">Imager</span><br />
      <span rel="v:wholistanalyst">Analyst</span>
  </div>
  <div>Working Memory
      <span rel="v:workingmemory">Low</span>
  </div>
</div>
```

3.2 Adaptation Mapping Layer

The Adaptation Mapping Layer is responsible for mapping the Users' Typologies of the User Profile Layer with the Smart Objects of the Web Content Layer. A Web Browser (Mozilla Firefox) Extension has been developed in order for the browser to recognize and implement the extended content objects (Table 5) and map them with the user's OCUM instance (Table 2).

Our main goal in this section is to show in a more detail how a Web browser should interpret the SmartObject of the RDFa schema (Table 4) and adapt the containing information based on the user's OCUM and consequently the abovementioned cognitive factors. Based on Tables 2 and 5 (Web Content Layer), the Web browser combines the user's OCUM with the containing information of the SmartObject entity, adapting the content.

The adaptation process involves the transformation and/or enhancement of a given raw Web-based hypertext content (provider's original content) based on the impact the specific human factors have on the information space [15] (i.e., show a more diagrammatical representation of the content in case of an Imager user, as well as provide the user with extra navigation support tools). Figure 2 shows the possible Web-based hypertext

content transformations / enhancements based on the mapping process that take place during adaptation process based on the influence of the human factors and the theory of individual differences.

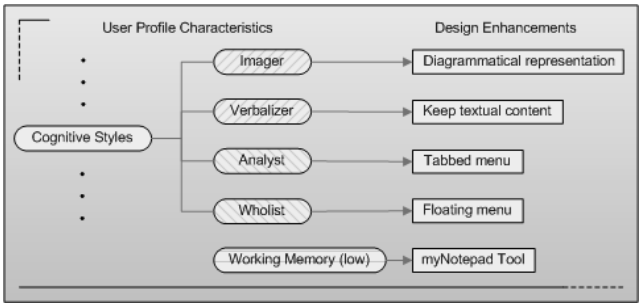


Figure 2. Hypertext Content Transformations / Enhancements

Based on figure 2, the meta-characteristics of a user profile are deterministic (at most 3); Imager or Verbalizer, Analyst or Wholist and Working Memory level (considered only when low).

For a better understanding, a user that happens to be an Imager gets a diagrammatical representation of the containing information of SmartObject. The "about" attribute is used by the Web browser to distinguish the logical meaning of a sentence when creating the diagrammatical representation. In other words, the "about" attribute is used for sub-elements of a SmartObject. As we will see furthermore, the "about" attribute is interpreted differently by the browser when the user types change. On the other hand, when a user is a Verbalizer (prefers text instead of diagrammatical representations), no changes are made to the containing content of SmartObject. Furthermore, if a user is an Analyst, the information will be enriched with a tabbed menu to be easier accessible. The menu will consist of the SmartObject sub-elements. Each sub-element along with the "title" property (see Table 5) is used in this case to create the tabbed menu with the title of each sub-element comprising an item of the menu.

Each sub-element is added to the tabbed menu and is used as a dynamic link to the containing information of the particular entity. The same logic of transformation is used when mapping the SmartObject with a Wholist user. In this case, a dynamic floating menu with anchors is created so to guide the users on specific parts into the hypertext content while interacting. Again, the sub-elements comprise the menu's items.

Finally, when users happen to have a low working memory level, the browser will provide them with the "myNotepad" tool (temporary memory buffer) for storing a section (sub-element's content) of the page and keep active information that is interested in until the completion of a cognitive task at hand.

3.3 Web Content Layer

The third layer of OAM; the Web Content Layer models any web content that comes from the provider. An RDFa schema (Table 4) has been designed that enables standard annotations in any XHTML Web-page, thus making structured data available for our framework's adaptation process, but also for any service or tool that supports the same standard. Table 5 shows an instance of the RDFa content model.

Table 4. Content Model RDFa Vocabulary

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF>
<rdfs:Class rdf:ID="SmartObject">
  <rdfs:comment>Represents an adaptive object based on users'
cognitive styles.</rdfs:comment>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/1999/02/22-
rdf-syntax-ns#Resource"/>
</rdfs:Class>
<rdf:Property rdf:ID="name">
  <rdfs:domain rdf:resource="#SmartObject"/>
</rdf:Property>
<rdf:Property rdf:ID="category">
  <rdfs:domain rdf:resource="#SmartObject"/>
</rdf:Property>
<rdf:Property rdf:ID="summary">
  <rdfs:domain rdf:resource="#SmartObject"/>
</rdf:Property>
<rdf:Property rdf:ID="title">
  <rdfs:domain rdf:resource="#SmartObject"/>
</rdf:Property>
<rdf:Property rdf:ID="content">
  <rdfs:domain rdf:resource="#SmartObject"/>
</rdf:Property>
</rdf>
```

The above vocabulary consists of a number of classes and properties which describe an adaptive object based on users' cognitive styles. The main class of this vocabulary is SmartObject that represents an adaptive web object. This class has the following properties: i) "name" property; the concept's name, ii) "category" property; the concept's category, iii) "summary" property; the summary description of the concept, iv) "title" property; the title of the concept's sub-element, and v) "content" property; the concept's sub-element content. The "about" property (Table 5) is used by RDFa to distinguish different sub-elements of the concept.

Table 5. RDFa Instance of a Content Object

```
<div xmlns:v="http://www4.cs.ucy.ac.cy/adaptiveWeb/rdf/#"
typeof="v:SmartObject">
    <span property="v:name">Sony</span>
    <span property="v:category">13' Laptop</span>
    <span property="v:summary">2.5GHz CPU Intel Core 2 Duo,
4GB RAM, 250GB HD</span>
    <div about="/sonyvaio/sz/memory">
        <span property="v:title">Memory Information</span>
        <span property="v:content">4GB Memory RAM, 250GB
Hard Disk</span>
    </div>
    <div about="/sonyvaio/sz/cpu">
        <span property="v:title">CPU Information</span>
        <span property="v:content">2.5GHz CPU Intel Core 2
Duo</span>
    </div>
</div>
```

In order to evaluate the system's performance as well as the impact of our model's dimensions into the information space, we have designed and authored an experimental environment in the

application field of eCommerce. The eCommerce (Web) environment that has been developed used the design and information (hypermedia) content of an existing commercial Web-site of Sony Style.com. This Web-site provides products' specifications of the Sony Company, and in general is very representative of the sites that we inspected in our high level analysis since it stands in between a serious layout and an aesthetically rich form of presentation. We have developed an exact replica of the Sony Vaio Notebooks section in sonystyle.com.

4. EXPERIMENTAL EVALUATION

The following section describes the experimental design and the results that support the notion of personalization and the use of OAM in generic Web-based hypertext environments.

4.1 Methodology & Design Implications

For the purposes of our research a within participants experiment was conducted, seeking out to explore if the personalized condition serves users better at finding information more accurately and fast. A pilot study that involved a between participants design demonstrated inconsistent effects, suggesting that a within subjects approach would yield more robust results.

The number of participants was 89; they all were students from the Universities of Cyprus and Athens and their age varied from 18 to 21, with a mean age of 19. They accessed the Web-based hypertext environments using personal computers located at the laboratories of both universities, divided in groups of approximately 12 participants. Each session lasted about 40 minutes; 20 minutes were required for the user-profiling process (real-time psychometric tests), while the remaining time was devoted to navigating in both hypertext environments, which were presented sequentially (as soon as they were done with the first environment, the second one was presented).

The hypertext content was about a series of Sony laptop computers: general description, technical specifications and additional information were available for each model. As stated in the introductory section, we considered that the original (raw) version of the environment was designed without any consideration towards cognitive style preferences, and the amount of information was so high and randomly allocated that could increase the possibility of cognitive overload. The personalized condition addressed these issues by introducing as personalization factors both cognitive style and working memory span.

The psychometric materials that were used are the following:

1. Cognitive Style: Riding's Cognitive Style Analysis, standardized in Greek, assessing the Imager/Verbalizer and Wholist/Analyst dimensions.
2. Working Memory Span: Visuospatial working memory test, examining participants' ability to temporarily store visual figures.

In each condition, users were asked to fulfill three tasks: they actually had to find the necessary information to answer three sequential multiple choice questions that were given to them while navigating. All six questions (three per condition) were about determining which laptop excelled with respect to the prerequisites that were set by each question. There was certainly only one correct answer that was possible to be found relatively easy, in the sense that users were not required to have hardware related knowledge or understanding.

As soon as users finished answering all questions in both conditions, they were presented with a comparative satisfaction questionnaire; users were asked to choose which hypertext environment was better (1-5 scale, where 1 means strong preference for environment A and 5 for environment B), regarding usability and user friendliness factors.

The dependent variables that were considered as indicators of differences between the two hypertext environments were:

1. Task accuracy (number of correct answers)
2. Task completion time
3. User satisfaction

At this point a few clarifications about the methodology are necessary:

1. Users did not know which the personalized condition was, nor were they encouraged to use any additional features.
2. To avoid training effects, half of the users received the raw condition first (considered as environment A), whilst the other half started the procedure with the personalized (again considered as environment A).
3. To avoid a possible effect of differences in difficulty of each set of three questions, they were alternated in both environments. Due to a design error, the division was not in half, but 53 participants received the first combination and 36 the alternated. However there was not observed any effect; all questions were proven of equal difficulty- to the extent that this is possible of course.

The within participants design, finally, allowed the control of differences and confiding variables amongst users.

4.2 Results

The most robust and interesting finding was the fact that users in the personalized condition were more accurate in providing the correct answer for each task. The same user in the raw condition had a mean of 1 correct answer, while in the personalized condition the mean rose to 1.9.

Since the distribution was not normal and the paired samples t-test assumptions were not met, Wilcoxon Signed Ranks Test was performed, showing that this difference is statistically significant at zero level of confidence ($Z= -4.755$, $p=0.000$). This is probably a very encouraging finding, implying that personalization on the basis of these factors (cognitive style and WMS) benefits users within an eCommerce environment, as long as there are some cognitive functions involved of course (such as information seeking).

Equally interesting is the fact that users in the personalized condition were significantly faster at task completion. The mean aggregated time of answering all three questions was 541 seconds in the raw condition, and 412 in the personalized. A paired samples t-test was performed ($t(88)=4.668$, $p=0.000$)

demonstrating significance at zero level of confidence. Again, this second dependent variable (time) shows that the personalized hypertext environment is more efficient.

As it concerns the satisfaction questionnaire, 31 users leaned towards the personalized environment, 38 had no preference while 20 preferred the raw. This descriptive statistic is merely indicative of whether participants would consciously observe any positive or negative effects of the personalized condition.

A considerable percentage leaned towards that condition (or at least users did not seem somehow annoyed by such a restructuring), but overall it cannot be supported that they were fully aware of their increase in performance, as shown by the abovementioned findings.

5. CONCLUSIONS & FUTURE TRENDS

The basic objective of this research paper was to introduce a cognitive approach to Web Personalization based on an ontology that contains users' cognitive factors. Accordingly, a human factors' ontology has been designed and developed using RDFa, and could be used in any Web-based application for returning an optimized adaptive result to the user. Their specific influence and the Web design enhancements and hypertext content transformations have been described and positively evaluated in the eCommerce domain.

It was clearly demonstrated that users' information finding was more accurate and efficient, both in terms of providing correct answers to the task questions and in task completion time. These findings reveal that our approach turned out to be initially successful, with a significant impact of human factors in the personalization and adaptation procedure of Web-based hypertext and hypermedia environments.

Even though the evaluation of the OCUM concept in the eCommerce domain is really encouraging for the validity and integrity of the relation within and between these cognitive dimensions and their effective impact in the information space, this ontology can only be considered as a proposal. Main goal is to initiate and drive this research to a concrete human factors ontology that can be used in any hypertext computer-mediated system enhancing one-to-one services delivery based on an efficient user-centric dynamic content reconstruction (adaptation).

Sub consequently, another major future step of our work, besides improving and extending the methodology of our experiments in a commercial / services Web environment, is the integration of emotional processing parameters, which involves the use of sensors and real-time monitoring of emotional arousal (Galvanic Skin Response and Heart Rate).

Finally, even if the Sony site is quite a representative Web-site of how information is distributed in the Web, further testing on various types of Web-sites and other computer-mediated platforms is required in order to establish a rigid connection between human factors and information processing in Web-based hypertext/hypermedia environments.

6. ACKNOWLEDGMENTS

The project is co-funded by the EU project CONET (INFSO-ICT-224053) and by the Cyprus Research Promotion Foundation under the project MELCO (ΤΠΕ/ΟΡ120/0308 (BIE)/14).

7. REFERENCES

[1] Germanakos, P., Samaras, G., and Christodoulou, E. Multi-channel Delivery of Services - the Road from eGovernment to mGovernment: Further Technological Challenges and Implications, 2005.

[2] Speretta, M., and Gauch, S. Personalized search based on user search histories. *In Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 622–628, 2005.

[3] Micarelli, A., and Sciarrone, F. Anatomy and empirical evaluation of an adaptive web-based information filtering system. *User Modeling and User-Adapted Interaction*, 14(2-3):159–200, 2004.

[4] Haav, H., and Lubi, T. A survey of concept-based information retrieval tools on the web. *In 5th East-European Conference*, pages 29–41, 2001.

[5] Trajkova, J., and Gauch, S. Improving ontology-based user profiles. *In Proceedings of the Recherche d'Information Assiste par Ordinateur*, pages 380–389, 2004

[6] Guarino, N., Masolo, C., and Vetere, G. OntoSeek: Content-Based Access to the Web, *IEEE Intelligent Systems* 14(3), 70-80, 1999.

[7] Knight, K., and Luk, S. Building a Large Knowledge Base for Machine Translation, *In proceedings of American Association of Artificial Intelligence Conf.*, 773-778, 1999.

[8] Labrou, Y., and Finin, T. Yahoo! As an Ontology – Using Yahoo! Categories to Describe Documents, *In proceedings of 8th Intl. Conf. on Information and Knowledge Management*, 180-187, 1999.

[9] Yahoo!, http://www.yahoo.com.

[10] Chowder, G., and Nicholas, C. Resource Selection in Café: an Architecture for Networked Information retrieval, *In proceedings of SIGIR'96 Workshop on Networked Information Retrieval*, Zurich, 1343 – 1355, 1996.

[11] Chower, G., and Nicholas, C. Meta-Data for Distributed Text Retrieval, *In Proceedings of SIGIR'97 Workshop on Networked Information Retrieval*, 317- 332, 1997.

[12] Pearce, C., and Miller, E. The Telltale Dynamic Hypertext Environment: Approaches to Scalability, *in Intelligent Hypertext: Advanced Techniques for the World Wide Web*, Vol. 1326, 109 – 130, 1997.

[13] Heflin, J., Hendler, J., and Luke, S. SHOE, A Knowledge Representation Language for internet Applications, *Technical Report CS-TR-4078,* Institute for Advanced Computer Studies, University of Maryland, College Park, 1999.

[14] Luke, S., Spector, L., Rager, D., and Hendler, J. Ontology-Based Web Agents, *In proceedings of First International Conference on Autonomous Agents*, 59 – 66, 1997.

[15] Germanakos, P., Tsianos, N., Lekkas, Z., Mourlas, C., and Samaras, G. Realizing Comprehensive User Profiling as the Core Element of Adaptive and Personalized Communication Environments and Systems, *The Computer Journal, Special Issue on Profiling Expertise and Behaviour*, Oxford University Press, doi:10.1016/j.chb.2007.07.010, 2008.