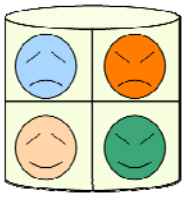


3rd Int'l Workshop on

"**Personalized Access, Profile Management, and Context Awareness in Databases**"

August 28, 2009

PersDB 2009

in conjunction with VLDB 2009

Lyon, France

3rd Int'l Workshop on "**Personalized Access, Profile Management, and Context Awareness in Databases**" (**PersDB 2009**)

Foreword

Emerging applications such as Web-based communities, wikis, social networks, mashups and folksonomies go beyond information access and dissemination to 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. 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 now of paramount importance in improving and personalizing user experience. The purpose of the PersDB workshop is to bring together database and information retrieval researchers and practitioners in order to study the new data management challenges which must take into account personal, social and contextual information about users.

We have received 18 papers spanning different topics including context-driven databases, recommendations, search and social networks. We accepted 6 papers, having a healthy acceptance rate of about 33%. We are grateful to all the authors who submitted papers and those who are attending the workshop.

We would also like to thank Laks Lakshmanan for accepting to be the keynote speaker, our panelists and our reviewers who did their best in delivering thorough reviews on time. Last but not least, we would like to thank Julia Stoyanovich for maintaining the workshop web site.

Sihem Amer-Yahia (Yahoo! Labs, USA) – Georgia Koutrika (Stanford University, USA)
PersDB'09 Program Chairs

3rd Int'l Workshop on "**Personalized Access, Profile Management, and Context Awareness in Databases**" (**PersDB 2009**)

Workshop Organization

Steering Committee

Tiziana Catarci	(*Universita di Roma "La Sapienza", Italy*)
Yannis Ioannidis	(*University of Athens, Greece*)
Timos Sellis	(*IMIS Institute, "Athena" Research Center, Greece*)

Program Committee Chairs

Sihem Amer-Yahia	(*Yahoo! Labs, USA*)
Georgia Koutrika	*(Stanford University, USA*)

Program Committee

Michael Benedikt	(*Oxford University, UK*)
Vassilis Christophides	(*ICS-FORTH, Greece*)
Paolo Ciaccia	(*University of Bologna, Italy*)
Irini Fundulaki	(*ICS-FORTH, Greece*)
Zoltan Gyongyi	(*Google Research, USA*)
Anne-Marie Kermarrec	(*INRIA, France*)
Werner Kiessling	(*University Augsburg, Germany*)
Alexandros Labrinidis	(*University of Pittsburgh, USA*)
Dongwon Lee	(*Penn State University, USA*)
Massimo Melucci	(*University of Padua, Italy*)
Bamshad Mobasher	(*DePaul University, USA*)
Atsuyuki Morishima	(*University of Tsukuba, Japan*)
Christos Papatheodorou	(*Ionian University, Greece*)
Evi Pitoura	(*University of Ioannina, Greece*)
Guillaume Raschia	(*Polytech Nantes, France*)
Yannis Stavrakas	(*IMIS Institute, "Athena" Research Center, Greece*)
Letizia Tanca	(*Polytecnico di Milano, Italy*)
Martin Theobald	(*Max-Planck-Institut fuer Informatik, Germany*)
Wolf Tilo Balke	(*University of Hannover, Germany*)
Alex Tuzhilin	(*Stern NYU, USA*)
Gerhard Weikum	(*Max-Planck-Institut fuer Informatik, Germany*)
Cong Yu	(*Yahoo! Research, USA*)

Table of Contents

Recommender Systems Revisited: from Items to Transactions (Invited Talk)

Laks V.S. Lakshmanan (University of British Columbia, Canada)

How Far Should We Personalize? (Panel)

Sihem Amer-Yahia (Y! Labs, USA) Yannis Ioannidis (University of Athens, Greece), Christian Jensen (Aalborg University, Denmark), Evi Pitoura (University of Ioannina, Greece), Elisa Quintarelli (Politecnico di Milano, Italy)

REGULAR PAPERS

CADS: a Collaborative Adaptive Data Sharing Platform

Vagelis Hristidis (Florida International University, USA), Eduardo Ruiz (Florida International University, USA)

Collaborative Ranking Function Training for Web Search Personalization

Giorgos Giannopoulos (National Technical U. of Athens, Greece), Theodore Dalamagas (IMIS Institute, "Athena" Research Center, Greece), Timos Sellis (IMIS Institute, "Athena" Research Center, Greece)

Domain Level Personalization Technique

Alessandro Campi (Politecnico di Milano, Italy), Mirjana Mazuran (Politecnico di Milano, Italy), Stefania Ronchi (Politecnico di Milano, Italy)

Guiding Personal Choices in a Quality Contracts Driven Query Economy

Huiming Qu (IBM Watson Research Center, USA), Jie Xu (U. of Pittsburgh, USA), Alexandros Labrinidis (U. of Pittsburgh, USA)

Context-Aware Recommender Systems: a Service-Oriented Approach

Sofiane Abbar (PRiSM Laboratory, Versailles University, France), Mokrane Bouzeghoub (PRiSM Laboratory, Versailles University, France), Lopes Stephane (PRiSM Laboratory, Versailles University, France)

You May Also Like Results in Relational Databases

Kostas Stefanidis (U. of Ioannina, Greece), Marina Drosou (U. of Ioannina, Greece), Evi Pitoura (U. of Ioannina, Greece)

Recommender Systems Revisited: from Items to Transactions (Invited Talk)

Laks V.S. Lakshmanan (University of British Columbia, Canada)

Abstract: Recommender systems have been extremely successful in reaching relevant content to users. Rather than rely on a static notion of content relevance to a user's query or a profile, they incorporate endorsements of items by other users and/or ratings provided by the same user on other items considered similar. In this talk, I will make a case for developing recommendation strategies and systems not just for recommending content items but for users performing transactions. Consider a social network where users register items (e.g., toaster, lawn mower) they are willing to give away to other users in exchange for items in their wish list which they have registered with the system. The idea is users either swap items or more generally exchange items in cycles. I will discuss the algorithmic challenges in developing strategies for recommending exchange transactions to users and present approximation as well as heuristic algorithms we have developed for solving this problem. I will also discuss the results of a detailed set of experiments we ran to gauge the performance of the various algorithms and conclude with interesting directions for future work.

How Far Should We Personalize? (Panel)

Sihem Amer-Yahia (Y! Labs, USA) Yannis Ioannidis (University of Athens, Greece), Christian Jensen (Aalborg University, Denmark), Evi Pitoura (University of Ioannina, Greece), Elisa Quintarelli (Politecnico di Milano, Italy)

Abstract: Given the proliferation of data and applications, different possibilities as well as different requirements for personalizing user experience emerge at various levels (e.g., content, UI, services, etc). For instance, people may like different services on their cell phone or in Facebook. They may be interested in different content depending on their location, task, preferences or group they belong to. Different presentation features serve different people better. For example, some users like lengthy explanations, others may want to see reviews. Endless personalization possibilities seem to exist in different applications, from personalized search and ads to personal mashups.

At the same time, there are several arguments against (over-) personalization. For example, if we do not have correct information about a user, personalization may hurt accuracy. In addition to gathering and maintaining a profile, on-the-fly personalization can be expensive. Over-personalization may lead to over-specialization. Making a recommendation of something a user would definitely like is valuable but what about serendipity and diversity? There is also a delicate balance in advertising between the accuracy of ads and their total irrelevance. From the publishers point of view, they would rather serve relevant content than not, from the users' perspective, more relevant ads may be annoying (e.g., a user announces the birth of a child in an email to friends, ads start suggesting where to buy baby stuff). With all these in mind, the panel's objective is to discuss when, what, how, and to what extent we should or should not personalize.

CADS: A Collaborative Adaptive Data Sharing Platform[1]

Vagelis Hristidis Eduardo Ruiz
School of Computing and Information Sciences
Florida International University
{vagelis,eruiz011}@cis.fiu.edu

ABSTRACT

Content management tools like Microsoft's SharePoint allow users of an application domain to share documents and tag them in an ad-hoc way. Similarly, Google Base allows users to define attributes for their objects or choose from predefined templates. This ad-hoc or predefined annotation of the shared data incurs problems like schema explosion or inadequate data annotation, which in turn lead to poor search and analysis capabilities.

We propose CADS, a Collaborative Adaptive Data Sharing platform, where the information demand of the community–e.g., query workload–is exploited to annotate the data at insertion-time. A key novelty of CADS is that it learns with time the most important data attributes of the application, and uses this knowledge to guide the data insertion and querying. In this position paper, we present the challenges and preliminary design ideas for building a CADS platform. We use the application of CADS on the Business Continuity Information Network (BCIN) of South Florida as a motivating example.

1. INTRODUCTION

There are many application domains where a community of users collaborate and share domain-specific information; for instance, news blogs, scientific networks, social networking groups, or disaster management networks. Current information sharing tools, like content management software (e.g., Microsoft's SharePoint), allow users to share documents and tag them in an ad-hoc way. Similarly, Google Base [14] allows users to define attributes for their objects or choose from predefined templates. For instance, when a user input a weather report on a hurricane, it would be nice to enter ⟨Storm category, 3⟩ or other such information. Even if the system allows users to arbitrarily annotate their data with such pairs, the users would probably be unwilling to do it since it requires considerable effort (inadequate data annotation). Further, the system would end up having thousands of different attribute names (schema explosion), where many share the same real life meaning, e.g., "Storm category", "Hurricane category", "Storm level". The above limitations make the analysis and querying of the data cumbersome. Users are mostly limited to plain keyword searches, with very few extra conditions like date and owner of document.

VLDB '09, August 24-28, 2009, Lyon, France.

[1]Partly supported by NSF grant IIS-0811922 and DHS grant 2009-ST-062-000016.

A recent line of work to the right direction is the pay-as-you-go querying strategy in Dataspaces, where users provide data integration hints at query time. However, there is no work that achieves integration and attribute-extraction of the data at insertion time, since a key assumption in previous works is that the data sources already exist. This assumption is generally not valid for collaborative data sharing platforms.

We propose CADS, a Collaborative Adaptive Data Sharing platform, which facilitates data annotation at insertion-time and leverages these annotations at query-time. CADS learns with time the information demand (query workload), which is then used create adaptive insertion and query forms.

Some of the collaborative data sharing applications that will benefit from a successful CADS platform are disaster management, corporate context management, news portals, social networking, and scientific collaboration.

Motivating scenario: Our motivating scenario is a disaster management situation, which was inspired by the experiences of the authors in building a Business Continuity Information Network [30] for disaster situations in South Florida. In this particular domain we have many users and organizations publishing and consuming information. For example, in a hurricane situation, local government agencies report shelters locations, damages in structures or structural warnings. Meteorological Agencies report the status of the hurricane, its position and particular warnings. Volunteers may share their activities and look for critical needs. Business owners may describe the status and needs of their stores and personnel.

The information produced and consumed in this domain is dynamic and unpredictable, and agencies have their own protocols and formats of sharing data, e.g., the Miami-Dade County Emergency Office publishes hourly document reports. Further, learning the schema from previous disasters is hard given that new needs, requirements and situations arise.

In Figure 1(a) we show a report extracted from the National Hurricane Center repository, which describes the status of a hurricane event in 2008, that is, the current storm location, wind speed, warnings, category, advisory identifier number and the date it was disclosed. Even though this is a text document, many ⟨attribute name, attribute value⟩ pairs, e.g., "Storm Category = 3" can be extracted, which could then improve the quality of searching through the database. For instance, Figure 1(b) shows three sample queries for which the report of Figure 1(a) is a good answer.

The goal of CADS is to allow the effortless sharing of documents like the one in Figure 1(a), while at the same time serving semi-structured queries like the ones in Figure 1(b).

The structure of the paper is as follows: In Section 2 we discuss the relationship of CADS to other research efforts. Section 3 presents the preliminary design of CADS. The research challenges of CADS are presented in Section 4 and we conclude in Section 5.

2. RELATED WORK

Dataspaces and Pay-as-you-go Integration: The integration model of CADS is similar to that of dataspaces [13], where a loosely integration model is proposed for heterogeneous sources. However, the semi-automatic annotation of data with metadata at insertion time is new to CADS. In CADS, the integration then occurs on this metadata. Another related data model is that of Google Base [14], where users can specify their own attribute/value pairs, in addition to the ones proposed by the system. However, the proposed attributes in Google Base are hard-coded for each item category (e.g., real estate property). In CADS, the goal is to "learn" what attribute/values to suggest. Pay-as-you go integration techniques like PayGo [25] and [22] are useful to suggest candidate matchings at query time. However, no previous work considers this problem at insertion time, as in CADS. The work on Peer Data Management Systems [16] is a precursor of the above projects.

```
ZCZC MIATCPAT2 ALL
TTAA00 KNHC DDHHMM
BULLETIN
HURRICANE GUSTAV INTERMEDIATE ADVISORY NUMBER 31A
NWS TPC/NATIONAL HURRICANE CENTER MIAMI FL   AL072008
600 AM CDT MON SEP 01 2008

EYE OF GUSTAV NEARING THE LOUISIANA COAST...HURRICANE FORCE WINDS
OVER PORTIONS OF SOUTHEASTERN LOUISIANA...
A HURRICANE WARNING REMAINS IN EFFECT FROM JUST EAST OF HIGH
ISLAND TEXAS EASTWARD TO THE MISSISSIPPI-ALABAMA
BORDER...INCLUDING THE CITY OF NEW ORLEANS AND LAKE PONTCHARTRAIN.
PREPARATIONS TO PROTECT LIFE AND PROPERTY SHOULD HAVE BEEN
COMPLETED.
A TROPICAL STORM WARNING REMAINS IN EFFECT FROM EAST OF THE
MISSISSIPPI-ALABAMA BORDER TO THE OCHLOCKONEE RIVER.
GUSTAV IS MOVING TOWARD THE NORTHWEST NEAR 16 MPH...26 KM/HR...AND
THIS MOTION IS EXPECTED TO CONTINUE FOR THE NEXT DAY OR SO WITH
SOME DECREASE IN FORWARD SPEED AND A GRADUAL TURN TOWARD THE WEST-
NORTHWEST ON TUESDAY.  ON THE FORECAST TRACK...THE CENTER WILL
CROSS THE LOUISIANA COAST BY MIDDAY TODAY.
MAXIMUM SUSTAINED WINDS ARE NEAR 115 MPH...185 M/HR...WITH HIGHER
GUSTS.  GUSTAV IS A CATEGORY THREE HURRICANE ON THE SAFFIR-SIMPSON
SCALE.
```

(a) Sample Document

```
Q1: Storm Name = 'Gustav' AND Warnings CONTAIN 'flood'

Q2: Storm Name = 'Gustav' AND Storm Category > 2

Q3: Document Type = 'advisory' AND Location = 'Louisiana'
    AND Date FROM 08/31/2008 TO 09/30/2008
```

(b) Sample Queries

Figure 1: Sample Document and Queries

Content Management products: Microsoft Sharepoint [26] and SAP NetWeaver [29] allow users to share documents, annotate them and perform simple keyword queries. Hard-coded attributes can be added to specialized insertion forms. CADS improves these platforms by learning the user information demand and adjusting the insertion and query forms accordingly.

Indexing, Provenance and Disagreement handling in data sharing environments: Data Ring [1] allows multiple peers to share content by declaratively defining the schema and capabilities in XML and leaving to the system the indexing and replication of the data. Orchestra [18] is also based on peer to peer schema integration and assumes the existence relational schemas. CADS maintains a centralized repository and hence these works cannot be directly applied.

Information Extraction (IE): We have witnessed considerable progress in IE, which has been recently partitioned to Closed and Open IE. [8] provides a recent overview of the IE area.

Closed IE requires the user to define the schema of the extracted tables along with rules to achieve the extraction. This is too much work for a user who inserts a document. The most relevant work in this area is the recent work of Jain et al. [21], which shows how IE systems can be combined to efficiently answer SQL queries on documents. However, they still assume that someone has created these IE systems for specific schemas.

Open IE [11] is closer to the needs of CADS. In particular, Open IE generates RDF-like triplets, e.g., (Gustav, is category, 3) with no input from the user. Next, we describe why Open IE is not appropriate for our needs, even though we plan to adapt some of their ideas. Open IE leads to a huge number of triplets, which prevent the successful execution of ⟨attribute name, attribute value⟩ structured queries and the suggestion of appropriate attributes to the users at insertion and query time in CADS.

The CIMPLE project [10, 6] uses IE techniques to create and manage data-rich online communities, like the DBLife community. In contrast to CIMPLE, where data is extracted from existing sources and a domain expert must create a domain schema, CADS is a data sharing environment where users explicitly insert the data and the schema automatically evolves with time. Nevertheless, the IE and mass collaboration techniques of CIMPLE can help in creating adaptive insertion forms in CADS.

Schema Evolution: Note that the adaptive annotation in CADS can be viewed as semi-automatic schema evolution. Previous work on schema evolution [3] did not address the problem of what attribute to add to the schema, but how to support querying and other database operations when the schema changes.

Query Forms: Existing work on query forms can be leveraged in creating the CADS adaptive query forms. [19] proposes an algorithm to extract a query form that represents most of the queries in the database using the "querability" of the columns. [20] extends this work discussing forms customization. [27] uses the schema information to auto-complete attribute or value names in query forms. A limitation of the above forms is that they do not consider the information demand or the entity matching uncertainties. In [6] keyword queries are used to select the most appropriate query forms.

3. CADS PRELIMINARY DESIGN

The CADS system has two types of actors: producers and consumers. Producers upload data in the CADS system using interactive insertion forms and consumers search for relevant information using adaptive query forms. In the rest of the paper the term data usually refers to a document; other types of data are also possible, but we focus on documents for simplicity. Figure 2 presents a typical CADS workflow. Figure 3 shows the possible components of the two major CADS modules, the Insertion and Query modules.

Insertion phase: The insertion phase begins with the submission of a new document to be included in the repository. After the user uploads the document, CADS analyzes the text and creates an adaptive insertion form with the set of the most probable ⟨attribute

name, attribute value⟩ pairs to annotate the new document. The user fills this form with the required information and submits it. The final stage consists of the storage of the associated document and metadata in the CADS repository.

Going back to our disaster management motivating scenario, Figure 4 presents the adaptive insertion form for the hurricane advisory document of Figure 1. After the user submits the document, the system analyzes the content, and finds that the following attributes are relevant: "Storm Name", "Storm Category", "Warnings". These attributes are added to a set of default attributes like: "Document Type", "Date" and "Location", which are basic metadata that a domain expert has provided for an application. The "Description" attribute is used to input the whole text of the document.

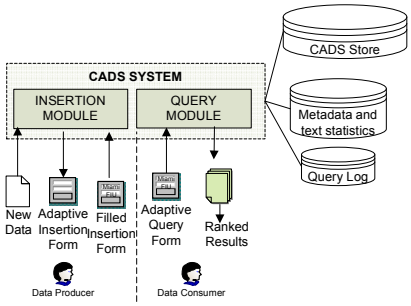


Figure 2: CADS Workflow.

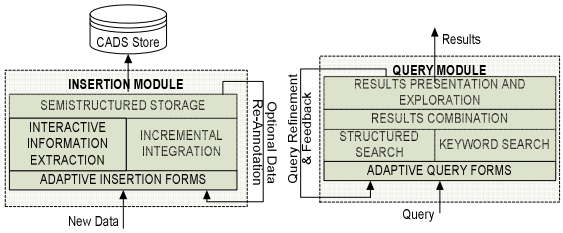


Figure 3: Architecture of Insertion and Query Modules.

In addition to extracting attribute names, the adaptive insertion form also extracts the attribute values by employing IE algorithms. A confidence threshold for the IE must be set. A lower threshold may bias the user and lead to errors in the data, whereas a high threshold may lead to many empty textboxes, which may frustrate the user. Ideally, the erroneous values are corrected and the missing attribute values are manually inserted by the user. This means that the quality of the data depends on the reliability of the users. User trust and anti-spam techniques must be considered for large-scale deployments of CADS.

As shown in Figure 4, attribute names and attribute values are presented as text boxes. If the user wants to associate more than one value to an attribute – e.g., multi-valued attributes like "Warnings"– then she can use the plus icon at the right to add attribute values. Each textbox has auto-completion capabilities, which exploit similar entries inserted before in the same attribute.

It is also important, to notice that a user can add new attributes, which are not suggested by the adaptive form. The form provides the option to do this task, in the spirit of the Google Base [14]. When the user specifies a new attribute, CADS will try to match it to existing attributes and show to the user a few matching options. The user can reject these suggestions and go ahead adding the new attribute. In this way, advanced users can collaborate for the schema construction.

Figure 4: Adaptive Insertion Form.

Query phase: In the query phase, the user is presented with an adaptive query form (Figure 5), which supports ⟨attribute name, attribute value⟩ conditions. Initially, before CADS has began learning the information demand through processing the query workload, the query form only specifies the default attributes (e.g., "Document type", "Date", "Location"). The user can specify additional ⟨attribute name, attribute value⟩ conditions. There is also a generic "Description" attribute where the user types keywords when she does not know how to put them in ⟨attribute name, attribute value⟩ conditions. The system discourages the user from just using the "Description" attribute, because this does not allow the system to learn the user information demand in a structured way, which in turn facilitates evolving the schema and performing schema mappings.

In some cases the conditions may trigger additional attributes recommendation, which CADS believes could be helpful for the user to further refine the query. For instance, if the user specifies the attribute "Storm Category" and previous users who specified "Storm Category" also specified "Wind Speed", then the adaptive query form will suggest to the user the attribute "Wind Speed". Further, if the attribute specified by a user is similar to another existing attribute, CADS will suggest a mapping between the two attributes, in the spirit of pay-as-you-go integration. Also, the system may suggest replacing the text in the generic "Description" attribute value with some ⟨attribute name, attribute value⟩ conditions.

When the user decides that her query form is complete, she submits the query. In this last phase CADS will find the most important pieces of data (e.g., document) for the query. The querying strategy must combine keyword search with uncertain structured query principles. The system returns a ranked list of the results, where the ranking is personalized. In order to personalize, CADS may assume that users generally look for similar items every time they search. A user profile may also be used. Also, note that CADS will typically return whole documents in the result. However, if the schema of the repository is mature and the query is selective, it is possible to return specific attribute values, in a way similar to the NAGA system [23]. The latter query result type is a possible future direction for CADS.

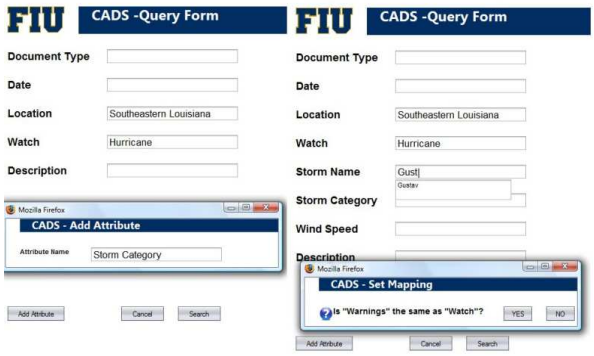


Figure 5: Adaptive Query Form.

In Figure 5 we show the progression of an adaptive query form in the disaster domain. In the left window we show the initial status of the query form. The generic form starts with some default attributes: "Document Type", "Location", "Description". The user is encouraged to specify other attributes, which do not only refine the query, but also help CADS learn the user information demand. For instance, in Figure 5 the user adds an attribute called "Storm Category" using the auxiliary window. Then, the form suggests to the user to also include the attributes "Storm Name" and "Wind Speed", which are correlated with "Storm Category" in the query workload. After that, the system tries to auto-compete the attribute value for "Storm Name" again using the past query workload. Finally, the system asks a pay-as-you-go schema mapping question: if "Warnings" is equivalent to "Watch", where the former is part of the existing schema (see Figure 4) and the latter is a user specified-attribute.

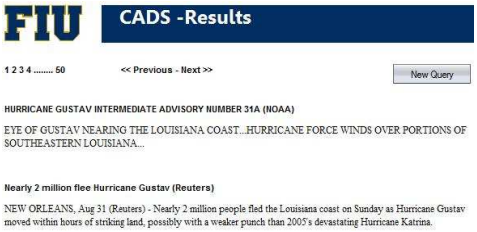


Figure 6: Query Results.

Figure 6 shows the results of the query. The document inserted in Figure 4 is the top result. Note that each result in the list may partially or fully satisfy the query, and is owned by a user. The trust degree of the owner for the querying user may be used as one of the ranking factors, in addition to factors like relevance and importance.

4. CHALLENGES AND RESEARCH DIRECTIONS

As mentioned in Section 2, the CADS platform can reuse much previous research on collaboration systems. However, many research pieces are missing, mainly regarding the algorithms behind adaptive insertion and query forms. We enumerate these challenges and preliminary ideas on how to address them.

Discover best ⟨attribute name, attribute value⟩ candidates for a newly inserted document: This line of research will decide what attributes the adaptive insertion form will suggest to the publisher (inserter). The following factors must be considered:

a) The *information value*, as specified by the past query workload *W*, which is related to the Value of Perfect Information in [22]. For instance, if the attribute "Storm Category" is used in many queries, then we may want to suggest it to a user that insert a document that contains the word "category". We will assign an information value $I^A(A_i, W)$ denoting how useful attribute A_i is, given W. A simple way to compute $I^A(A_i, W)$ is to count the number of queries in W that specify A_i. If the user has already specified some conditions in the adaptive query form, our algorithm will use their correlation to A_i in W. We will create a probabilistic model based on the Probabilistic Information Retrieval (PIR) ideas of our previous work [7]. The estimation of $I^A(A_i, W)$ should also exploit the associations in the *CADS Graph* (Figure 7), which connect groups, users, and data. In particular, we assume that the data *d* submitted by a user *u* is of more interest to users *x* who are closely associated to *u* on *G*, e.g., through common groups. We can weigh the queries in the workload according to their relevance to *u*.

b) The *confidence* that an attribute A_i is relevant for a to-be-inserted document *d*. The rationale of this factor is that we do not want to suggest to the user an attribute just because it is popular in the query workload, if this attribute does not have a good chance to be relevant to *d*. A_i may be relevant to *d* if we discover (e.g., through IE algorithms) that A_i appears in *d*, or if another attribute A_j appears in *d*, which is highly correlated to A_i. The correlation will be computed based on *W*. Hence, the attribute confidence $C^A(A_i,d,W)$ depends on all A_i,*d*,W. We need to adapt Information Extraction (IE) algorithms to compute $C^A(A_i,d,W)$ for document data, which are the main focus of CADS. In particular, we should leverage the work on Open IE [11] to extract triplets of extracted data, and then apply thesaurus and ontological knowledge (e.g., WordNet), as [12]. Further, producers have an inclination to publish all their documentation with a similar structure, which the system could learn. This observation came from our interaction with the Miami-Dade County Emergency Office, where the published reports typically have a common header and structure.

Matching of attribute names and attribute values across queries and inserted documents: Given that CADS is an open system, it is possible that different users use different names or structures to represent the same concept. We should consider matchings between attribute names or between attribute values. The matching between different schemas is a well known problem [28, 33, 24, 34] with various proposed solutions based on analysis of the data content or the schema properties. A main principle in CADS is that integration will occur in a semi-automatic way at both insertion and query time. In order to minimize the user involvement, previous schema matching and entity disambiguation [35] methods must be adapted in order to create a good ranking of the candidate matchings at insertion and query time, and present the user with a very small set of disambiguation questions. The work on pay-as-you-go integration [22] is an excellent starting point.

As in the case of attributes suggestion described above, candidate matchings $M(r,s)$ are ranked based on two factors: The Information Value $I^M(M,W)$ and the confidence $C^M(M,d,W)$. The $I^M(M,W)$ measure is based on the following intuition: If a user

submits attribute r, and s has a high information value in W, then the matching between r and s will also have high $I^M(M,W)$. The queries in W may be weighted based on their relevance to user u who submits *d*, as described above.

The $C^M(M,d,W)$ consider not only the workload W but also the inserted data *d*. Our problem is different from previous pay-as-you-go integration projects [25,22], because the integration occurs at insertion time and not only at query time. Hence, the confidence of a confirmed matching is much more credible than in the case of query-time integration, because the publisher confirms matchings of her own data, and not of possibly other sources. Further, the candidate matchings are ranked based on a combination of the data annotations and the raw data content (e.g., text of document). We could employ a learning algorithm, similar to the edge weight learning algorithm in [5], to weigh these factors based on the past user selections.

Another difference is that CADS can leverage the community links and data associations, represented at the CADS Graph (Figure 7) to guide the matching process. A matching algorithm can be created by expanding the similarity flooding idea [24] to operate on the more complex CADS Graph. In [24], the two candidate schemas were represented by a set of table schemas. In contrast, the CADS Graph also contains data instances, users and groups. Recent work [15, 32, 9] performed relevance ranking of the nodes for query answering purposes. We must adapt these works to do similarity ranking – e.g., combine FolkRank [15] with Similarity Flooding [24].

Storage of annotation data: It is challenging to efficiently store the documents and their metadata (extracted data), in a way that CADS will scale to thousands of users and millions of shared data. As different documents will have different attributes, this information could be very sparse, so a relational model could be very inefficient to implement. Further, attributes are dynamically added to the system. A more promising alternative is a triplet model, which represents **(*d*,e,v)** facts where ***d*** is a document id, **e** is the attribute name or predicate and **v** is the value of the attribute. The storage of triples is well studied in RDF systems [31]. It has also been studied in clinical management systems [9].

Discover best conditions to suggest in adaptive query forms: We need to exploit past query workload, historic data and user interactions, to create the best adaptive query form for a user. A good adaptive query form will allow the user, who is not aware of the structure of the data in the repository, to better express her query.

There has been significant work on user-friendly query interfaces (query forms) to express database queries, as discussed in Section 2. These works assume a well-defined schema (relational or XML) and a valid instance. In contrast, in our problem we have a set of data pieces, submitted by different users, with imprecise annotation schemata. Further, the content (e.g., text) of the data and the user associations must be considered.

For every candidate attribute A_i (or value) in the workload W, we will assign a relevance score $R(A_i,u,W,Q)$, given the user u and the current state of the adaptive query form Q, i.e. the already specified attributes and values. Then, the top-k ranked attributes will be suggested to the user, where k is a small number depending on the size of the screen real estate of the adaptive query form. The relevance has the following components:

a) The *user affinity*, that is, the relevance degree of user u to the attribute A_i. For that, we will create the query workload graph G_W, where the attributes of the past queries of each user will be connected to the user, and the users will be connected to each other through common group nodes, as in Figure 7. That is, G_W will have user, attribute and group nodes. We can use the idea of SimRank proposed in [2].

b) The correlation between Ai and the selected conditions Q. We can employ the ideas from our work on ranking SQL query results [7], where association rule mining is used to compute the attribute correlations. These techniques must be modified to account for the user associations. In particular, we will weigh the queries in W according to the relevance of the user who submitted each query to u.

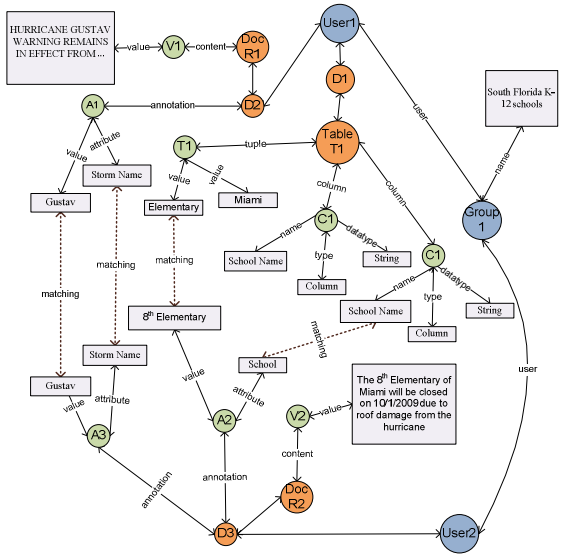


Figure 7: CADS graph

Ranking query results: After the user submits the query to the system, CADS must use a strategy to rank the data *d* in the repository *D*. Recent work [15, 32, 4] on querying tagged Web pages is an excellent starting point on how tags and users can be leveraged to query Web pages. They generally model users, pages and tags as a tripartite graph and propose adaptations of the PageRank algorithm. However, these works view the queries and the annotations (tags) as lists of keywords, that is, they do not consider any structure on the query or the annotations. Further, they only use the tags of the pages and not the page content. Instead, our ranking algorithms will exploit both the annotation structure and the raw content of the data.

A unique characteristic, which has not been studied before, is that a data piece *d* may be relevant to a query q either based on its annotations or based on its raw content. This introduces semantic and performance challenges. How should the annotations be weighed vs. the content to achieve a relevance score for *d*? How can we create efficient hybrid algorithms that avoid querying both the annotations and the content of *d*?

In terms of ranking semantics, if the query contains both structured conditions ("city"="Miami") and plain keywords

("flood"), a possible strategy is to use the structured query as filter and use the keyword query for ranking. This strategy is simple to implement, but assumes that the structure part of the database is complete and correct. As some documents are not appropriately annotated, the system needs a more intelligent strategy that takes into account the probabilistic nature of the annotations, that is, we are not sure if an annotation is missing because it is not appropriate for *d*, or because the publisher did not spend the time to add. Nevertheless, annotations should generally be viewed as more important than the raw text, because they can provide a Boolean match to the query. In contrast the text only provides a fuzzy matching. Hence, a query strategy may primarily rank the results *d* by how much the annotations of *d* match the query conditions, and secondarily on the IR-style relevance of the query to the text of *d*. Learning algorithms must be created to balance these factors based on the user feedback, i.e., results click-thru.

To address the problem of the probabilistic nature of the annotations, previous work on ranking under uncertainty [17] must be adapted for the hybrid filter/ranking model of CADS querying. This incurs efficiency and scalability issues, which require smart execution algorithms to achieve real-time responses.

5. CONCLUSIONS

We proposed CADS, a Collaborative Adaptive Data Sharing platform, which is a next-generation data sharing platform where the annotation and integration occur at both the data insertion (production) and querying (consumption) actions. A key goal of CADS is to leverage the information demand to create adaptive insertion and query forms. We believe that CADS has a great potential to improve many collaboration environments, and hence it is worthwhile to pursue research directions that will allow the realization of CADS.

6. REFERENCES

[1] Serge Abiteboul, Neoklis Polyzotis, The Data Ring: Community Content Sharing, In CIDR, pages 154-163, 2007

[2] G. Jeh, and J. Widom. SimRank: a measure of structural-context similarity. ACM SIGKDD international Conference on Knowledge Discovery and Data Mining. KDD 2002

[3] J. Banerjee, W. Kim, H. Kim, and H. F. Korth. 1987. Semantics and implementation of schema evolution in object-oriented databases. SIGMOD Rec. 16, 3 (Dec. 1987), 311-322.

[4] Heymann, P., Koutrika, G., and Garcia-Molina, H. Can social bookmarking improve web search?. International Conference on Web Search and Web Data Mining. WSDM '08.

[5] Ramakrishna Varadarajan, Vagelis Hristidis, Louiqa Raschid. Explaining and Reformulating Authority Flow Queries. IEEE ICDE 2008

[6] E. Chu, A. Baid, X. Chai, A. Doan, J. Naughton. Combining Keyword Search and Forms for Ad Hoc Querying of Databases, *SIGMOD-09*.

[7] S. Chaudhuri, G. Das, V. Hristidis, G. Weikum: Probabilistic Information Retrieval Approach for Ranking of Database Query Results. ACM Trans. Database Syst (TODS) 31, 3 (Sep. 2006)

[8] Michael J. Cafarella, Jayant Madhavan, Alon Y. Halevy: Web-scale extraction of structured data. SIGMOD Record 37(4): 55-61 (2008)

[9] R. S. Chen, P. Nadkarni, L. Marenco, F. Levin, J. Erdos, and P. L. Miller. Exploring Performance Issues for a Clinical Database Organized Using an Entity-Attribute-Value Representation. J. Am. Med. Inform. Assoc. 7: 475-487, 2000.

[10] A. Doan et al. Community Information Management, IEEE Data Eng. Bulletin, Probabilistic Databases, 29(1), 2006.

[11] O. Etzioni, M. Banko, S. Soderland, and *D.* S. Weld. Open information extraction from the web. Commun. ACM 51, 12 (Dec. 2008), 68-74.

[12] Fernando Farfán, Vagelis Hristidis, Anand Ranganathan, and Michael Weiner. XOntoRank: Ontology-Aware Search of Electronic Medical Records. ICDE 2009

[13] Michael J. Franklin, Alon Y. Halevy, David Maier: From databases to dataspaces: a new abstraction for information management. SIGMOD Record 34(4): 27-33 (2005)

[14] Google Base. http://www.google.com/base, 2009

[15] Andreas Hotho and Robert Jäschke and Christoph Schmitz and Gerd Stumme. Information Retrieval in Folksonomies: Search and Ranking. ESWC 2006, (4011):411-426, 2006

[16] A. Y. Halevy, Z. Ives, *D.* Suciu, I. Tatarinov, Schema mediation in peer data management systems, ICDE 2003

[17] M. Hua, J. Pei, W. Zhang, and X. Lin. Ranking queries on uncertain data: a probabilistic threshold approach. ACM SIGMOD 2008.

[18] Zachary G. Ives et al. The ORCHESTRA Collaborative Data Sharing System. SIGMOD Record 37(3): 26-32 (2008)

[19] Magesh Jayapandian and H. V. Jagadish. Automated Creation of a Forms-based Database Query Interface. In VLDB, 2008.

[20] Magesh Jayapandian and H. V. Jagadish. Expressive Query Specification through Form Customization. In EDBT, 2008.

[21] Alpa Jain, AnHai Doan, Luis Gravano: SQL Queries Over Unstructured Text Databases. ICDE 2007: 1255-1257

[22] Shawn R. Jeffery, Michael J. Franklin, Alon Y. Halevy: Pay-as-you-go user feedback for dataspace systems. SIGMOD Conference 2008: 847-860

[23] Gjergji Kasneci, Fabian M. Suchanek, Georgiana Ifrim, Maya Ramanath, Gerhard Weikum: NAGA: Searching and Ranking Knowledge. ICDE 2008: 953-962

[24] Sergey Melnik, Hector Garcia-Molina, Erhard Rahm: Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. ICDE 2002: 117-128

[25] J. Madhavan, S. R. Jeffery, S. Cohen, X. Dong, *D.* Ko, C. Yu, A. Halevy: Web-scale Data Integration: You can only afford to Pay As You Go. CIDR 2007. Asilomar, California, 2007

[26] Microsoft Sharepoint. http://www.microsoft.com/Sharepoint/ 2009

[27] A Nandi, HV Jagadish. Assisted querying using instant-response interfaces. Demo. Proceedings of the 2007 ACM SIGMOD

[28] E. Rahm, and P. Bernstein. A survey of approaches to automatic schema matching. The VLDB Journal 10, 4 (Dec. 2001), 334-350.

[29] SAP NetWeaver Capabilities - Content Management https://www.sdn.sap.com/irj/sdn/nw-cm ,2009

[30] K. Saleem, S. Luis, Y. Deng, S-C. Chen, V. Hristidis, T. Li. Towards a Business Continuity Information Network for Rapid Disaster Recovery. 9th Annual International Conference on Digital Government Research 2008

[31] Kevin Wilkinson , Craig Sayers , Harumi Kuno, Dave Reynolds. Efficient RDF Storage and Retrieval in Jena2 In Proc. of SWDB'03, VLDB 2003

[32] M. Bender et al. Exploiting social relations for query expansion and result ranking. In Data Engineering for Blogs, Social Media, and Web 2.0, ICDE 2008 Workshop

[33] AnHai Doan, Pedro Domingos, and Alon Y. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. SIGMOD 2001

[34] Wensheng Wu, Clement Yu, AnHai Doan, and Weiyi Meng. An interactive clustering-based approach to integrating source query interfaces on the deep web. SIGMOD 2004.

[35] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In SIGKDD, 2002

Collaborative Ranking Function Training for Web Search Personalization

Giorgos Giannopoulos
School of ECE
NTU Athens -
IMIS Institute
"Athena" Research Center
Greece
giann@dblab.ece.ntua.gr

Theodore Dalamagas
IMIS Institute
"Athena" Research Center
Greece
dalamag@imis.athena-innovation.gr

Timos Sellis
School of ECE
NTU Athens -
IMIS Institute
"Athena" Research Center
Greece
timos@imis.athena-innovation.gr

ABSTRACT

In this paper, we present a framework for improving the ranking function training and the re-ranking process for web search personalization. Our method is based on utilizing clickthrough data from several users in order to create multiple ranking functions that correspond to different topic areas. Those ranking functions are combined each time a user poses a new query in order to produce a new ranking, taking into account the similarity of the query with each of the topic areas mentioned before. We compare our method with the traditional approaches of training one ranking function per user, or per group of users and we show preliminary experimental results.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval—*Relevance feedback, Search process, Clustering*.

General Terms

Algorithms, Experimentation, Measurement

Keywords

Search engine, ranking, training, clickthrough data, relevance judgement, clustering

1. INTRODUCTION

Lately, a lot of effort has been put in finding ways to personalize web search results. Several models for ranking function training have been proposed [21, 3, 6] and a great number of studies on user search behavior and feedback have been performed [20, 1, 7, 12].

WOODSTOCK '97 El Paso, Texas USA

To the best of our knowledge, most approaches of learning ranking functions for search personalization, utilize machine learning techniques that train a ranking function for a user or a group of users with similar search behaviour. So, no matter how diverse the search topics of a user, or a group of users sharing a common ranking function training, are, the re-ranking/personalization of the search results is based on the same model.

However, it is often the case that a user searches in more than one different areas of interest that could lead to completely different ranking function training models. Consider for example the following search scenario. A phd student working on information retrieval (IR) issues would like to search for papers related to clickstream data, machine learning techniques and ranking. Her search is usually based on keywords related to IR. Since ACM is a well-known digital library, she would probably prefer clicking on results from that data source. So, a proper model for this search behavior would train a ranking function that favors results whose titles have high textual similarity with the query and results containing the word "acm" in their title or url.

However, the user would also like to search for information about a new cellphone she would like to buy. She would use the brandname of the cellphone as well as words like "review" or "hands on" as keywords. Also, she would probably click on results coming from forums where users discuss their experience/opinion about the cellphone and/or on video results which would present the phone's functionality. So a model suitable for this kind of search would favor results containing the word "forum/blog" in their url and video results.

Thus, it is evident that training a single ranking function per user or per group of users with similar search behavior does not capture the diversity in topics areas that are of user interest.

Our approach. In this work, we address the aforementioned problem. We present a framework that creates multiple ranking functions, each one corresponding to a different topic area, which are finally combined to produce a final score for each search result. The main idea is that a ranking function is not trained for a single user, neither a group of users with similar search behavior. On the contrary, one ranking function is trained per *topic area*. That is:

1. We gather clickthrough data from several users and

extract triplets of the form: (*query*, *result list*, *clicked results list*).

2. Then, we cluster the clicked results based on their textual similarity and for each result cluster C_i, we train a different ranking function F_i using Ranking SVM. So, when a user poses a new query, each ranking function F_i will give a different rank r_{ij} for each query result j.

3. We compute the similarity between the query and each cluster C_i, and we represent it with a weight w_i.

4. We then exploit w_is to produce a final ranking list of results to present to the user.

Outline. In Section 2 we discuss the background issues related to ranking function learning. In Section 3, we present our method for adjusting the ranking function training and result re-ranking according to the content of each new query. In Section 4 we present a preliminary experimental evalution. Section 5 presents the related work and, finally, Section 6 conludes and discusses further work.

2. RANKING FUNCTION TRAINING

In this section, we present background information on ranking function training. We outline the steps that comprise this process, including: a) extraction of relevance relations, b) extraction of features, and c) training of ranking function.

Relevance relation extraction. Based on the results of a query, relevance relations can be extracted from either *absolute* or *relative* preferences. An absolute preference suggests that a search result is either relevant or irrelevant to a query. A relative preference suggests that a search result is more relevant to a query than another result.

The extraction of those relations is mainly based on the results the user viewed or clicked. There are many approaches for extracting relevance relations. For instance, in [7, 11], where an approach that uses relative preferences is adopted, relevance relations are extracted by taking into consideration the relative position of a pair of results in the initial ranking. For example, a clicked result c is considered more relevant than all non-clicked results which are placed higher than c in the ranking list. An example of absolute preference is presented in [14]. In this work, if a result is clicked, then it is considered relavant to the query.

Usually, every query-result pair is assigned a score or a label, named its *relevance judgement*, that denotes the degree of relevance between the result and the query. The set of query-result pairs, together with the corresponding relevance judgements, comprise the first component that is input to the training process. The second component involves feature extraction from the query-result pairs.

Feature extraction. Every query-result pair is represented by a feature vector which quantifies the matching quality between the query and the result. There exists a great variety of features that can be used in this process. Content-based features, which can be extracted from the title, the body, the anchor and the url of a result, can be used to estimate content similarity between the query and the result [9]. Some other features are based on hyperlink information (i.e., pagerank values) [2], or on specific information of the results such as the domain of the url or the rank of the result in several search engines [7]. Also, such features may incorporate statistical information over user behaviour, e.g., deviation from average time spent for viewing pages [1].

Ranking function training. Ranking function training aims at assigning a proper weight for each feature used in feature vectors. Those weights indicate which features are more important for ranking the search results in the context of the particular training process. Several training methods have been proposed, such as Ranking SVM [21], RankNET [3], RankBoost [6] as well as tuning methods for the aforementioned techniques [4].

Figure 1 shows the overall training process. Once the ranking function is trained, the results for each new query are re-ranked according to scores produced by the ranking function. Those scores are calculated utilizing the weights found in the previous step and the feature vector of each new search result. In the example, three labels are used for relevance judgement: *irrelevant*, *partially relevant* and *strongly relevant* results respectively. For each query-result pair viewed or clicked by the user, these relevance judgements can be extracted implicitly as described previously in this section. For example, if a result is not clicked at all, it is *irrelevant*. If it is clicked only 1-2 times, it is *partially relevant*, while if it is clicked more times, then it is *strongly relevant*.

We should note that, regardless of the approach adopted, there has to be a specific representation of the relevance relations, so that they can be used as input to a ranking function training system. In this paper, we adopt the representation format of SVM$^{\text{light}}$[1], which is a popular tool that implements Ranking SVM. Also, we should point out that our method sticks to the process described above, proposing a different perspective in the training and re-ranking phases.

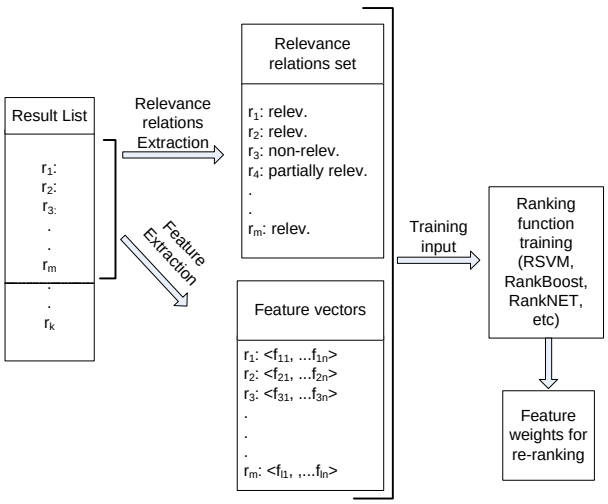


Figure 1: Ranking function training.

3. COLLABORATIVE TRAINING

In our method, we consider clickthrough data of the form: (*query*, *result list*, *clicked results list*). We are based on Joachims model [7] according to which a clicked result is more relevant to the query than all non-clicked results in higher rank. These relevance relations produce relevance

[1]http://svmlight.joachims.org/

judgments to train a ranking function and adapt ranking to user needs.

Our aim is to have multiple ranking functions, each one trained per *search topic*. We identify search topics by clustering user clickthrough data, and more specifically results clicked by users. Our approach involves the following steps:

- First, we perform clustering on the clicked results of all queries posed by users. We choose to cluster the clicked results instead of the queries to capture user behaviour because we consider them more informative about the user information need. According to the well known example, if a user searches for "jaguar", some results will be related to the animal and some others to the car. If we use the query text and cluster the queries, the information about which of these two concepts the user is actually interested in will be lost. On the contrary, if we cluster the clicked results using, i.e., title and abstract text, we will be able to capture her actual interest.

  After the clustering is performed, groups of results with similar content are formed. We call these groups *topic clusters*.

- We need a mechanism to calculate the similarity of every new query with each of the extracted clusters. For this reason, we use titles and abstracts of all (clicked) results belonging to a cluster as the textual representation of this cluster. We build an inverted file index to be able to calculate the similarity of the query with the textual representation of each cluster.

- We need to create one ranking function model per topic cluster. To achieve that, we use as training input only clickthrough data related to the corresponding topic cluster.

- This final step deals with the re-ranking of the results, utilizing the clusters created previously. When a new query is posed, we re-rank its results using all available ranking function models producing N different rankings, where N the number of topic clusters. In order to compose the final ranking, we combine those rankings taking into account the similarity of the query with each topic cluster.

Next, we describe in detail the aforementioned steps.

3.1 Clustering of Search Results

Consider a feature space ϕ of n terms, $\phi = \{t_1, t_2, \ldots, t_n\}$, where n is the total number of distinct terms in all clicked results for all queries of all users. We represent each result by a feature vector $v_i = \{wtd_{i1}, wtd_{i2}, \ldots, wtd_{in}\}$, where $wtd_{ik} = tf_{ik} * log(N/df_k)$, tf_{ik} is the frequency of occurrence of term t_k in document i, N is the number of results, and df_k is the number of results that contain the term t_k. We should note that, in our approach, we take into consideration the title and the abstract of each result in order to extract those features.

After extracting the features, we cluster the results having similar content. We use a partitional clustering method that utilizes repeated bisections [16, 17]. This method has been shown to have excellent performance when dealing with document collections [16]. In the following, we give an overview of the clustering method.

All documents (i.e., search results) are initially partitioned into two clusters (i.e., bisected). Then, one of these clusters is selected and is further bisected. This process is repeated until we get the desired number of clusters. In each step, the selection of the cluster to be bisected and the bisection itself, is done in such a way that the bisection optimizes the value of a clustering criterion function.

The criterion function used to form the clusters aims at maximizing the following quantity:

$$\sum_{i=1}^{k} \sqrt{\sum_{v,u \in S_i} sim(v,u)}$$

where k is the number of clusters, S_i is the set of documents of cluster i, and $sim(v,u)$ is the similarity value between documents v and u in S_i.

We employ the cosine similarity as the metric that compares two documents. We apply the cosine similarity function on the documents' feature vectors as shown in the following Equation:

$$sim(v,u) = \frac{\sum_{i=1}^{n}(wtd_{vi} \times wtd_{ui})}{\sqrt{\sum_{i=1}^{n} wtd_{vi}^2} \times \sqrt{\sum_{i=1}^{n} wtd_{ui}^2}}$$

Following this process, we obtain N *topic clusters*, with each one containing similar clicked results.

3.2 Cluster Indexing

Given a topic cluster C_i, we extract the text from the titles and abstracts of all the results it contains. We regard this text as the cluster's textual representation. In this way, we obtain a collection of "documents" representing the topic clusters. Then, we build an inverted file index on these documents using the Lucene[2] IR engine. The index is then used to calculate the similarity of any new query (see Section 3.4) with each cluster:

1. We pose the query to the Lucene search engine.
2. The search engine uses its own scoring function to calculate the simirarity of each indexed document with the query.
3. A list of documents is returned, along with a score that indicates how similar to the query they are.
4. The scores are normalized so that the sum of all scores equals to 1.
5. We assign each normalized document score to the corresponding topic cluster.

3.3 Ranking function training

For every extracted topic cluster C_i we train a different ranking function F_i as described in Section 2. The training is based on Joachims' model [7, 11] using ranking SVMs. Relevance judgments and feature vectors are used as input to the SVMs. The relevance judgements are produced by the users' clicks. To construct feature vectors, we have implemented the following features:

1. Textual similarity between query and (title, abstract, URL) of the result. The similarity is computed with 3 different types: tfidf, BM25 and Lucene scoring function. This results to 9 different similarity features.

[2]http://lucene.apache.org/

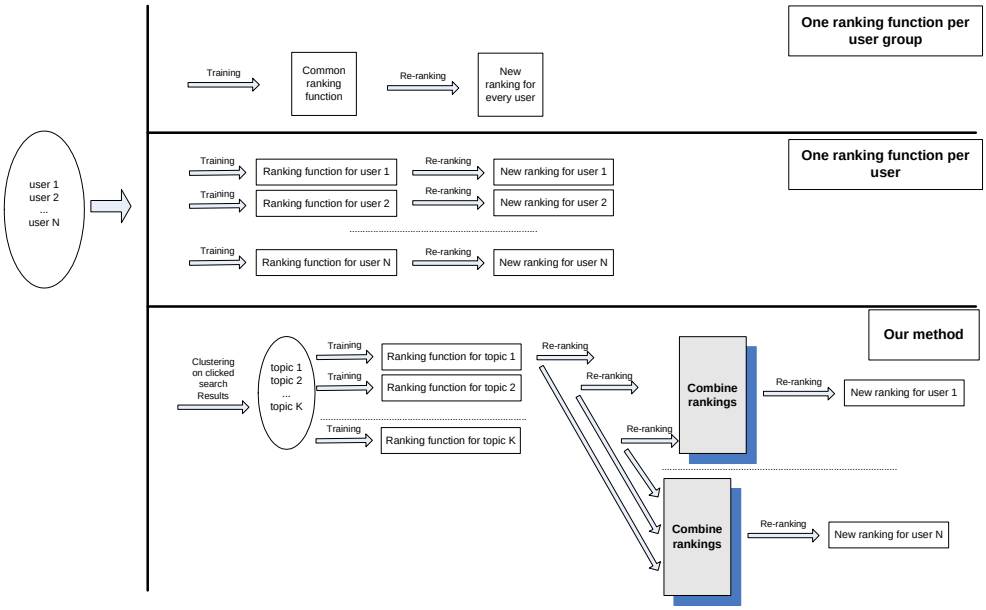


Figure 2: Training and re-ranking approaches

2. Domain of the result (.com, .edu, etc): 73 boolean features (i.e., exist or not).

3. Rank of the result in Google search engine.

4. Special words found such as "blog", "forum", "wiki", "portal", etc, found in the result's title, abstract or url. Each word corresponds to 3 feature values. These values depend on the similarity scores of this word on the result's title, abstract and url.

5. URL suffix (.html, .pdf, .ppt): boolean features.

6. 100 most frequent words found in all result documents of all previous searches. Each word corresponds to 3 feature values. These values depend on the similarity scores of this word on the result's title, abstract and url.

As training input for the Ranking SVM, we use only clickthrough data related to the corresponding topic cluster. That is, we regard only query-result pairs, where results belong to the same cluster. This process results to the creation of N ranking function models, where N is the number of topic clusters.

3.4 Re-Ranking

All steps described in the previous sections regard the training phase. This section deals with re-ranking which is performed after training is over. Re-ranking involves the following issues:

- When a user poses a new query q, its similarity score w_{qi} with every topic cluster C_i is calculated as described in Section 3.2.

- Then, using each ranking model F_i, we produce N different rankings R_{qi} for query q (r_{qij} the rank of result j) according to model F_i, corresponding to cluster C_i.

- The final rank for each result j for query q is given by the type:

$$rank(q, j) = \sum_{i=1}^{N} w_{qi} r_{qij}$$

4. EXPERIMENTAL EVALUATION

In this section, we present a preliminary experimental evaluation of our method. First, we present our experimental dataset.

4.1 Dataset

In order to obtain clickthrough data, we set up a logging mechanism over Google Search Engine. The application is able to keep track of the queries posed by the users, the result list returned from Google (i.e., title, abstract and URL), and the result URLs that were clicked by the users. Also, it records auxiliary information, such as (a) the IPs, so that we are able to distinguish the users, and (b) the exact date and time of queries and result clicks, in order to be able to divide the clickthrough dataset into training and test dataset.

We asked from 10 users, phd students and researchers from our lab, to search on Google for information relevant to a given set of topics for a two-months period. The search topics were the following: *Gadgets, Cinema, Auto & Moto, Life & health*, and *Science*. The users were asked to select 1 to 3 out of these topics and focus their search mainly on those, without, however, to restrict them from searching for information relevant to the other topics. During this period of time, we got clickthrough data for 671 queries. We used 75% of clickthrough data as the training dataset (501 queries), and the 25% as the test dataset (170 queries).

4.2 Preliminary Results

We evaluate the effectiveness of the following approaches for training ranking functions:

T1: Training one ranking function per user.

T2: Training one ranking function per group of users who are expected to have similar search behaviors.

T3: Collaborative ranking function training (our approach).

Our dataset is based on implicit user feedback, and not on data explicity judged. Thus, there are only a few relevance judgements for the results of each query. So, it was not possible to use evaluation metrics such as Precision and Mean Average Precision, since those metrics require a great number of query results in the test dataset to have relevance judgements. In our case, for most queries we have judgements (i.e., clicks by the users) for 1 or 2 results.

However, what we were able to do, was to compare the ranks coming from the different ranking function training approaches. Specifically, we executed 3 rounds of ranking function training and re-ranking (one for each approach T1, T2 and T3), and compared the ranks of clicked results in the new re-ranked result lists.

The results are presented in Table 1. Since our method depends on clustering quality, we also give results obtained varying the number of clusters created (see Section 3.1). The values presented in column "T3-T1" ("T3-T2") are the average differences in the ranks of the clicked results in the new re-ranked result lists produced by approaches T1 and T3 (T2 and T3). For example, the value 3 shows that our approach T3 moves the clicked results 3 positions higher, on average, compared to T1.

num of clusters	T3-T1	T3-T2
5	3	27
10	-14	11
15	-11	10
20	-9	15
25	-12	12

Table 1: Average differences of clicked results ranks in the new re-ranked result lists

Determining the proper clustering arrangement, we can improve the re-ranking process for personalization of search results. We can see that for the first clustering arrangement (5 clusters) our method outperforms the other two.

Some interesting results are presented in Table 2 where each cell shows the percentage of clicked results belonging to each cluster for each user. For example, 31% of results clicked by user 2 belong to the topic area represented by cluster 1.

We observe that there are users who searched only in one topic area (users 1 and 10), users who searched mainly in two areas (users 2, 5 and 7) and users who searched in many areas (users 3, 4, 6, 8 and 9). This observation supports our intuition that we should train and use more than one ranking models even for the same user.

We also observe that users 1, 3, 4 and 7 dedicated a respectable percent of their searches in the topic area represented by cluster 4, and, similarly, users 5, 7, 8 and 10 for cluster 5, and users 5, 6, 8 and 9 for cluster 3. This supports our intuition that we should train and use ranking models by combining clickthrough data from more than one users.

Finally, we should note here that we do not aim at comparing the efficiency of different training and ranking models, but at examining how our method improves the effectiveness of a given model (in our case Ranking SVM), by creating multiple ranking functions and combining them according to the user search needs.

5. RELATED WORK

In what follows, previous approaches regarding the problem of increasing the quality of ranking function training and re-ranking are presented. In [18] the authors utilize concept hierarchies like ODP[3] to categorize queries and build user profiles. They, then, use collaborative filtering techniques to re-rank query results based on those profiles. Our method differs in that we do not construct different profiles for each user, but utilize the information from several users to create topic clusters, in which users participate.

The approaches described in [12] and [10] are based on rank promotion of results having low rank. The initial ranking presented to users is slightly altered using several strategies, so that more results are judged by the users. As a result, a better ranking function training is achieved.

Finally, clustering is exploited in SVM Ranking methods, but to achieve different goals. In [8], clustering of results is utilized in order to refine a large training data set, by rejecting those data that are not necessary for the SVM training phase. Also, in [5], clustering is performed on the ranking functions learned from training data taken from different users. In [19] we used clustering to increase the training input, inferring relevance judgements for unjudged results.

6. CONCLUSION

In this paper, we presented a methodology for improving the quality of ranking function training. Performing clustering on clickthrough data involving search results clicked by the users, we find groups of similar results that represent topic areas. Based on those groups, we train multiple ranking functions each one corresponding to a different topic area, which are finally combined to produce a final score for each search result.

The intuition behind our method is that a user may exhibit more than one search behaviors, which cannot be represented by a single ranking model. However, if we exploit multiple instances of a ranking model, that are collaboratively trained for all users, better results can be achieved. Thus, we suggest an approach where one ranking function is trained per topic area, contrary to approaches where a ranking function is trained per user, or for a group of users with similar search behavior. The experiments show that our approach gives better results compared to those approaches.

Our work is in progress and this is a first-cut approach to this problem with preliminary experimental results. Evidently, there is room for improvements and expansions. First, we plan to perform more extended experiments with larger datasets in order to obtain more reliable results and study the effects of clustering in the topic area detection. We also plan to study whether classification techniques using pre-defined concept hierarchies (ODP) can help in detecting more appropriate topic areas. Finally, we will study more sophisticated mechanisms for inferring topic areas. Now, we cluster clicked results based on their textual content similarity. We believe that we could achieve better results by performing clustering exploiting the similarity values of feature vectors for each query-result pair.

[3]http://www.dmoz.org/

user	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
1	0	0	0	100	0
2	31	67	0	2	0
3	6	11	16	67	0
4	11	48	0	41	0
5	0	0	32	4	64
6	10	4	72	7	7
7	2	0	4	24	70
8	9	7	24	5	55
9	27	4.5	55	4.5	9
10	2	2	2	0	94

Table 2: Percentage of clicked results belonging to each cluster (topic area) for each user.

7. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26, 2006.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30:107–117, 1998.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[4] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 186–193, 2006.

[5] J. Diez, J. J. del Coz, O. Luaces, and A. Bahamonde. Clustering people according to their preference criteria. *Expert Systems with Applications: An International Journal*, 34:1274–1284, 2008.

[6] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal of Machine Learning Research*, 4:933–969, 2003.

[7] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

[8] X. Li, N. Wang, and S.-Y. Li. A fast training algorithm for svm via clustering technique and gabriel graph. In *Proceedings of the International Conference on Intelligent Computing*, 2007.

[9] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, 2007.

[10] S. Pandey, S. Roy, C. O. J. Cho, and S. Chakrabarti. Shuffling a stacked deck: the case for partially randomized ranking of search engine results. In *Proceedings of the 31st international conference on Very large data bases*, pages 781–792, 2005.

[11] F. Radlinski and T. Joachims. Query chains: Learning to rank from implicit feedback. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248, 2005.

[12] F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 570–579, 2007.

[13] S. E. Robertson. Overview of the okapi projects. *Journal of Documentation*, 53(1):3–7, 1997.

[14] G.-R. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 118–126, 2004.

[15] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342, 2001.

[16] Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, 2004.

[17] Y. Zhao, G. Karypis, and U. Fayyad. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168, 2005.

[18] U. Rohini and V. Ambati. Improving Re-ranking of Search Results Using Collaborative Filtering. *Information Retrieval Technology, Third Asia Information Retrieval Symposium, AIRS 2006*, pages 205–216, 2006.

[19] G. Giannopoulos, T. Dalamagas, M. Eirinaki and T. Sellis. Boosting the ranking function learning process using clustering. *10th ACM International Workshop on Web Information and Data Management (WIDM 2008)*, pages 125–132, 2008.

[20] S. Fox, K. Karnawat, M. Mydland, S. Dumais and T. White. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems (TOIS)*, 23(2):147–168, 2005.

[21] R. Herbrich, T. Graepel and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classiers, MIT Press*, Pages: 115-132, 2000.

Domain Level Personalization Technique

Alessandro Campi
Politecnico di Milano
Dipartimento di Elettronica e Informazione
via Ponzio 34/5, 20133, Milano, Italy
campi@elet.polimi.it

Mirjana Mazuran
Politecnico di Milano
Dipartimento di Elettronica e Informazione
via Ponzio 34/5, 20133, Milano, Italy
mazuran@elet.polimi.it

Stefania Ronchi
Politecnico di Milano
Dipartimento di Elettronica e Informazione
via Ponzio 34/5, 20133, Milano, Italy
ronchi@elet.polimi.it

ABSTRACT

We propose a novel technique for web search personalization that exploits the clustering of the results of web searches. Our approach is based on an automatic characterization of the user search history through the collection of semantic domains and web sources chosen by the user. The semantic domains are the terms extracted directly from the clusters contents which describe the main topics covered by the involved documents. The web sources are the web roots of the urls of the documents. The idea is that a user submits a query to a general purpose search engine and then selects clusters or documents from the resulting list. In the first case we can assume that, for that particular query, the user is interested in the topics covered by the selected clusters. We use these information to construct a user profile by assigning the clusters semantic domains to the query submitted by the user. In the second case we keep trace of the relevance and reliability of web sources of the selected documents, by assigning them to the submitted query.

1. INTRODUCTION

Recent researches investigate the ability of current search engines to address the diverse goals that people have when they submit the same query to a search engine. The potential value of personalizing search results is quantified and great variance was found in the results that different individuals rated as relevant for the same query. The analysis suggests that while search engines perform well in ranking results to maximize global happiness, they do not do a very good job for specific individuals [22]. In recent years a lot of research work was devoted to overcome this limitation by proposing ways to make the search engine aware of the context of its users in order to adapt the search results with respect to it. By learning the context of the users it is possible to personalize the search engine result list and to provide more valuable results to user queries.

In this paper, we present a novel approach of web search personalization, that exploits the clustering of the resulting documents in order to create a complete user profile based on a *characterization* of the user past search history. This operation is realized through the usage of two different information: the *semantic domains*, and the *web sources*. In particular, for *semantic domain* we mean a term s_d that express a "user-specific meaning" of a generic query term q_t (for example if q_t = *Java* we can have s_d = "language" or s_d = "island"). The aim of these domains is to disambiguate a general query term w.r.t. the user preferences. On the other hand, for *web source* we mean the root of the sources considered reliable for the user (i.e. `www.java.com`) w.r.t her/his past searches and document choices. This mechanism of user profile construction has two important characteristics: it is automatically realized without any explicit effort from the user or other contributions from external sources (i.e. ontologies, thesaurus, etc.), and it works at the level of web sites, differently from most of other personalization techniques that act directly on specific documents.

The final goal of our contribution is to learn the preferences of a user in order to support a personalized ranking of future results (both at document and at cluster level), and to provide the user with additional queries which may be interesting with respect to her/his profile.

2. RELATED WORK

Our work is part of a long research stream on personalization in Web searches. The work in [15] proposes a technique to map a user query to a set of categories, which represent the user's search intention. The set of categories are used to disambiguate the query terms while a user profile is learned from the user history and a category hierarchy respectively. This work is different from our in the usage of a predefined set of domains documents are classified in. The Inquirus 2 project [21] uses context information, currently in the form of a category of desired information ("personal bookmarks", "research papers", etc.). Differently from our approach this work does not exploit clustering of pages and the categories used to classify pages are not based on the semantics of their contents.

The work in [3] focuses on re-ranking the Web search output according to the cosine distance between each URL and a set of terms describing user's interests. An evolution of the work is [4] which proposes to improve Web queries by expanding them with terms collected from each user's personal information repository (personal collection of text documents, emails, Web pages, ...). An alternative approach is to compute a topic-oriented PageRank [9], in which PageRank vectors biased on each of the main topics of the Open

VLDB '09, August 24-28, 2009, Lyon, France

Directory were initially calculated off-line, and then combined at run-time based on the similarity between the user query and each of the topics. In this work the set of topics is predefined, while in our approach topics are dynamically built. The idea is extended in [17] by distributing the PageRank across the topics it contains in order to generate topic-oriented rankings. An algorithm that avoids the massive resources needed for storing one personalized PageRank vector per user by precomputing them only for a small set of pages and then applying linear combination is proposed in [11]. Machine Learning on the past click history of the user [19] can be used in order to determine topic preference vectors and then apply Topic-Sensitive PageRank.

Alternative approaches are based on the idea of expanding the user query with new terms related to the input keywords. Such relationships are usually extracted from large scale thesauri, as WordNet [7, 20, 12, 16]. The study in [5] proposes a new method for query expansion based on query logs. The central idea is to extract probabilistic correlations between query terms and document terms by analyzing query logs. These correlations are then used to select high-quality expansion terms for new queries. [13] proposes a method to generate refinements or related terms to queries by mining anchor text for a large hypertext document collection. Search results can be grouped based on different query meanings [10]. It is done using WordNet to determine the meanings of each query term and merging similar senses with a grouping algorithm that employs a combination of categorization and clustering techniques.

Another important research stream is devoted to exploit query expansion in order to obtain a better formulation of user query. The main idea is that useful information can be extracted from the relevant documents returned for the initial query. A literature review of the beginnings of this research topic is described in [6]. [2] introduces the usage of clusters asking users to choose relevant clusters, instead of documents, thus reducing the interaction. Summarization can be used to extract informative sentences from the top-ranked documents and uses these sentences to expand the user query [14]. RIB (Recommender Intelligent Browser) [25] categorizes Web snippets using socially constructed Web directory such as the Open Directory Project. Snippets are clustered to improve the categorization. The created user profile is used to propose search results to users. [8] proposes a method to personalize a user's experience within a folksonomy using clustering. In particular, unsupervised clustering methods are used for extracting commonalities between tags, and the discovered clusters are used as intermediaries between a user's profile and resources in order to tailor the results of search to the user's interests. These approaches are different from ours in the usage of a predefined set of categories: terms are generated to classify pages on-the-fly using the page contents. Similarly, [24] explores an approach that focuses on the "social annotations of the web" which are annotations manually made by normal web users without a pre-defined formal ontology. Compared to the formal annotations, although social annotations are coarse-grained, informal and vague, they are also more accessible to more people and better reflect the meaning of the web resources from the users' point of views during their actual usage of the web resources. The derived emergent semantics are used to discover and search shared web bookmarks.

"London"	
cl.1 Wikipedia (0.669) { <LLondon, 1.0> ; <wikipedia, 0.5> ; <free, 0.5> ; <encyclopedia, 0.33> ; <city, 0.25> }	*London wikipedia*
cl.2 London Trinity College (0.606) { <London, 1> ; <college, 0.5> ; <university, 0.5> ; <king, 0.3> ; <colleges, 0.2> }	*London college*
cl.3 Google News London (0.598) { <London, 1> ; <news, 0.421> ; <city, 0.16> ; <guide, 0.16> ; <times, 0.16> }	*London news*
...	...
cl.8 Travel Guide (0.474) { <London, 1> ; <guide, 0.49> ; <travel, 0.25> ; <theatre, 0.14> ; <entertainment, 0.13 }	*London travel guide*
cl.9 Airport (0.451) { <airport, 1> ; <London, 0.57> ; <international, 0.43> ; <advice, 0.28> ; < parking, 0.28> }	*airport LLondon international*
...	...

Figure 1: Results presentation at clusters level.

3. OUR APPROACH

Our personalization proposal is based on the idea of tracking the choices performed by the user on the list of results obtained after she/he submits a query. The proposed approach exploits Matrioshka [1]. It allows users to submit queries to search engines (such as Google, Yahoo, Google Scholar) in order to obtain clustered and labeled results. Search engines return results representing Web pages characterized by title, url and snippet. When one submits a query to a selected search engine, the resulting documents are clustered using the Lingo clustering algorithm [18]. The result is presented as a list of labeled and ranked clusters. Labels are built considering the set of most relevant terms extracted from titles and snippet of the clustered documents. The retrieved significant terms are also used in order to define a disambiguated query for each cluster. The role of the new queries is to allow deepening the search with more specific queries.

The result of a query submission is shown in Figure 1. In the first column are presented the retrieved clusters. In particular, each cluster c_i is characterized by a label l_i, an overall weight ow_i of relevance of the cluster general content w.r.t. the submitted query, the set of weighted semantic domains $S_i = \{sd_{i1}, ..., sd_{ij}, ..., sd_{in}\}$ associated with the cluster, and their corresponding weights wsd_{ij}. In the second column are presented the disambiguated (expanded) queries extracted from each resulting cluster.

The personalization process is based on the tracking of the choices performed by the user on the result list of clustered documents. Our objective is to learn the user preferences, and to use them in order to:

1. rank the clusters of a search result list, showing in the first positions the clusters containing more interesting contents from the user viewpoint;

2. rank the documents contained in each cluster, in order to show in the first positions inside each cluster the documents belonging to the sources the user preferred in previous search processes;

3. recommend a set of terms taken from her/his profile for the expansion and the specialization of her/his original queries.

To achieve our goals, we consider two types of interaction between the user and the list of the retrieved clusters:

1. *Query execution*: the user chooses to submit a query either by typing it in the search engine or by choosing it in the set of the disambiguated queries associated to clusters or by choosing one of the recommended terms.

2. *Click tracking*: the user chooses a specific document or a specific cluster, in order to explore it.

Both actions indicate that the user is interested in a certain topic. We use these information to create a *user profile*. In particular, we store:

1. the *web sources* ($W = \{ws_1, ws_2, \ldots, ws_n\}$) of the chosen documents (such as `www.expedia.com`, etc). Such information are used to assign a reliability degree to each source. Intuitively, users choose to explore the pages of sources considered reliable, we increase the ranking of the sources users seem to prefer;

2. the *semantic domains* ($S = \{sd_1, sd_2, \ldots, sd_m\}$) considered interesting by the user. Such information are represented by the keywords corresponding to the clusters of interest.

These information are closely related to a specific submitted query and build the history we want to learn.

Differently from the techniques commonly used in personalization, we do not track the information related to a specific document (e.g. url, title, snippet, ...) because such information have a too fine granularity for the personalization process. Furthermore, the usage of the information of single documents often represents a limitation. For example:

1. it is possible to find urls representing different sections of the same web site. To keep trace of all of them, we need to store different information about the same document, obtaining duplicates and wasting space;

2. if we track the exact document url, the stored information could be used in order to represent only that document, and, in particular, only that single page of the entire web site. So, if we retrieve correlated (contiguous) pages as result of the same search, we can bring in the first positions of the resulting list only the pages that has a traced url;

3. the results of different submissions of the same query on a search engine are often different, especially if they were obtained by submissions very distant in time, even if the document sources are usually the same.

Hence, our general assumption is that a user is not actually interested in a specific page of a web site, but she/he is more interested in obtaining documents from a reliable "favorite" source, as first results of a specific query.

So, let us suppose that a computer scientist submits the queries "java" and "apple". We assume that s/he is first interested in obtaining results from her/his more reliable sources such as `www.java.com` and `www.apple.com` respectively, rather then from other sites such as `www.sun.com` or `www.allaboutapple.com/`, or from incoherent sources such as `www.bali-travel-online.com` (that considers java as an island) and `www.applefruit.it` (that considers apple as a fruit). The discrimination on the individual documents is less important under the same query.

4. HISTORY MATRICES

To store all the information needed by the personalization process we propose a data model based on the use of matrices. In particular, we conceive three types of matrices, the User Profile matrix, the Source Reputation matrix and the Source Annotation matrix. In this section we give the details about each of them.

4.1 User Profile matrix

The User Profile matrix P is constructed from the user query terms and the semantic domains associated with the clusters the user has clicked on. Given a query Q composed of a set of terms $\{qt_1, qt_2, ..., qt_n\}$, every time the user clicks on a cluster c, identified by the semantic domains in S, it means that the user is interested in the semantic domains in S, with respect to the query Q. This consideration is used to update the User Profile matrix which associates a degree of relevance between query terms and semantic domains. In fact, the matrix represents the function $\mathcal{P} : \{qt_i, sd_j\} \rightarrow w_{ij}$, which associates with each query term qt_i and each semantic domain sd_j a weight w_{ij} which indicates how relevant sd_j is with respect to qt_i.

In particular, the weight w_{ij} of a couple $\{qt_i, sd_j\}$ is the average of the weights of sd_j in their original clusters. Such average is computed with respect to the number of times the user has clicked on a cluster containing sd_j among its semantic domains. Moreover, in the matrix each query term is coupled with the number of times the user has asked for a query containing that term, and each semantic domain is coupled with the number of times the user has been interested in that semantic domain (which means s/he has clicked on a cluster containing those keywords). The matrix is represented as:

	$\{sd_1,f_1\}$	...	$\{sd_m,f_m\}$
$\{qt_1,f_1\}$			
...		w_{ij}	
$\{qt_n,f_n\}$			

where:

$$w_{ij_{new}} = \frac{w_{ij_{old}} * f_{j_{old}} + w_{j_c}}{f_{j_{new}}},$$

is computed incrementally using the weights of the semantic domain (sd_j) obtained for the same query term (qt_i) but in various submission, and $f_{j_{new}} = f_{j_{old}} + 1$ is the new frequency value associated with sd_j.

Let us suppose the user queries "London" for the first time and obtains as a result the following two clusters (for each cluster is indicated its set of semantic domains and the weight each of them has in the cluster):

```
c1: {(hotel,0.4),(travel,0.6)}
c2: {(theater,0.2),(movie,0.3),(entertainment,0.5)}
```

the user then clicks on the second cluster in order to examine its content. This action would result in the following User Profile matrix:

	{theater,1}	{movie,1}	{entertainment,1}
{london,1}	0.2	0.3	0.5

Let us now suppose the user queries "London hotels" and obtains the following clusters:

```
c3: {(flight,0.4),(travel,0.6),(entertainment,0.3)}
c4: {(economic,0.2),(booking,0.3)}
```

and then clicks on the first cluster. The new User Profile matrix is[1]:

	{th,1}	{mo,1}	{en,2}	{fl,1}	{tr,1}
{london,2}	0.2	0.3	0.4	0.4	0.6
{hotel,1}	0	0	0.3	0.4	0.6

The weight of the semantic domain "entertainment" for the query term "london" is evaluated as $(0.5 + 0.3)/2 = 0.4$.

Note that if we compute the logic AND between the query term "london" row and query term "hotel" row, we can easily identify which semantic domains are related to the multi-word query "london hotel" (in this case: flights, travel and entertainment). Thus, this matrix is suitable for both single-term and multi-term queries.

4.2 Source reputation matrix

The Source Reputation matrix R is constructed from the query terms and the web sources of the documents the user has clicked on. Given a query Q, we assume that every time the user clicks on a document, s/he is interested in that specific source of information with respect to the query. This consideration is used to update the source reputation matrix which associates a frequency between query terms and web sources. In fact, the matrix represents the function $\mathcal{R} : \{qt_i, ws_k\} \rightarrow f_{ik}$, which associates with each query term qt_i and each web source ws_k a frequency f_{ik} which is the number of times the user has clicked on a document whose source is ws_k, with respect to the query. The matrix is represented as:

	ws_1	...	ws_k
qt_1			
...		f_{ik}	
qt_n			

Let us suppose the user queries "London" and then clicks on the following documents:

```
d1: en.wikipedia.org/wiki/London
d2: www.visitlondon.com/
d3: www.expedia.co.uk/
```

Subsequently, the user queries "London hotels", and then clicks on the following documents:

```
d4: http://www.visitlondon.com/accommodation/
          hotels/
d5: http://www.expedia.co.uk/daily/holidays/
          packages.aspx?rfrr=-13006
```

The final Source Reputation Matrix is[2]:

	wiki	visitlondon	expedia
london	1	2	2
hotel	0	1	1

[1]In the following table we use some shortcuts: th=theater, mo=movie, en=entertainment, fl=flight, tr=travel

[2]In the following table we use some shortcuts: wiki=en.wikipedia.org, visitlondon=www.visitlondon.com, and expedia=www.expedia.co.uk

As for the previous matrix P, also the structure of R allows the identification of relevant multi-terms query sources through the application of the AND operator on the involved query terms rows. Thus `en.wikipedia.org` is not a relevant source for the query "London hotels" because the AND between 1 (for the corresponding "london" row) and 0 (for the corresponding "hotel" row) returns 0. Together with the matrix we retain a value called MAX_R devoted to store the maximum number contained in the matrix R. This is useful for normalization purposes.

4.3 Source Annotation matrix

The Source Annotation matrix A is constructed as the multiplication between the data in the User Profile matrix P and the data in the Source Reputation matrix S, such as $A = P^T \times S$. In this way, the source annotation matrix is used to create a relation between the web sources and the semantic domains. In fact, the matrix represents the function $\mathcal{A} : \{sd_j, ws_k\} \rightarrow w_{jk}$, which assigns to each semantic domain sd_j and each web source ws_k a weight w_{jk} that indicates how much a semantic domain is relevant with respect to a web sources. The matrix is represented as:

	ws_1	...	ws_k
sd_1			
...		w_{jk}	
sd_m			

Let us consider the two examples presented in the previous sections. The resulting Source Annotation Matrix is:

	wiki	visitlondon	expedia
theater	0,2	0,4	0,4
movie	0,3	0,6	0,6
entertainment	0,4	1,1	1,1
flight	0,4	1,2	1,2
travel	0,6	1,8	1,8

5. USER PROFILE CONSTRUCTION AND MAINTENANCE

As already introduced in Section 3, the information we consider in our personalization technique can be collected and updated when:

- the user submits a query;
- the user chooses a certain cluster. We can deduce that the set of semantic domains corresponding to that cluster are of interest to the user, with respect to the query;
- the user chooses a certain document. We can deduce that the web source of the document is considered reliable by the user w.r.t the query.

When the user submits a query, we use the historical data to rank the results of the query and to suggest the user new possibly interesting queries. On the other hand, when the user performs some choices on the results of a query, we use such information to update the historical data.

In the following we give a more detailed description of all the operations we considered interesting for the user profile construction and maintenance.

5.1 Query execution

The execution of a query happens in three different scenarios: (1) the user submits her/his query to the search engine; (2) the user submits one of the possible disambiguated queries built considering her/his profile; (3) the user submits one disambiguated query from the search results list and has therefore submitted the new request.

The query is then first preprocessed by the engine which means that all stop-words are removed and all terms are stemmed. The result is the set of significant terms contained in the query, $Q = \{qt_1, qt_2, \ldots, qt_n\}$. Each term is then used to update the history matrices P and R. In particular, $\forall$ terms $qt_i \in Q$:

- if qt_i is already contained within the P matrix, its frequency is increased by one and the corresponding value of f is updated
- otherwise, it is added to it with $f = 1$
- if qt_i is not contained within the R matrix, it is necessary to add a new line representing the new term

5.2 Choice of a cluster

The user chooses a particular cluster c_j and explores the documents contained in the cluster. In this scenario, the set of semantics domains associated with the cluster is used to update the User Profile matrix P. In particular, $\forall$ term $qt_i \in Q$, the semantic domains $S = \{sd_1, sd_2, \ldots, sd_m\}$ associated with the cluster c_j, are eventually added as columns of P (if they don't already exist in it). Moreover, $\forall w_{ij} \in P$ such as $i = 0, ..., n$ are indexes of query terms which originated the selected cluster:

- if $w_{ij} \neq 0$ in P, its value is updated with the sd_j weight in the selected cluster, as described in 4.1.
- if $w_{ij} = 0$ in P, $w_{ij} = sd_j$ weight.

5.3 Choice of a document

The user chooses a specific document d_h contained in a cluster c_l. In this scenario, the web source wd_j of the document is used to update the Source Reputation matrix R.

In particular, $\forall$ term $qt_i \in Q$:

- a corresponding row in the R table is created (if it is not already present in the matrix)
- a column for web source ws_j is added (if it is not already present in the matrix)
- the frequency f_{ij} is evaluated for the web source ws_j. In particular:
  - if the web source is already contained in the set, its frequency f_{ij} is added by 1.
  - if the web source is not contained, it is added to it with $f_{ij} = 1$.

Note that in this scenario the semantic domains associated to the explored cluster are not taken into account. In fact they were already stored when the user clicked on the cluster label, looking for interesting documents.

In order to avoid the explosion of the dimensions of the matrices and guarantee the scalability of the approach we use classical techniques for efficient sparse matrices management and cleaning techniques to delete not useful data.

6. RANKING OF THE RESULTS

Let us suppose the user submits a query. The result of the query is a set of clustered documents which need to be ranked before being presented to the user. In order to rank the results, we access the historical matrices with the aim of giving a higher rank to the information "similar" to the previously browsed. Algorithm 1 shows how to rank the clusters resulting from a web search given the user profile matrix P, the set of clusters C and the query Q submitted by the user.

Algorithm 1 Rank-Clusters (P, C, Q)

1: **for all** clusters $c_j \in C$ **do**
2: K=\{semantic domain $sd_k \in c_j$\}
3: for all qt consider the set ROW of rows of matrix P corresponding to the query terms
4: build the set COL of columns having only non zero values in the cells in the rows in R
5: insert in S_Q the terms corresponding to COL
6: $X = \{wsd_{ij} : sd_{ij} \in (K \cap S_Q)\}$
7: $CW = \frac{\sum_X x}{|K \cap S_Q|}$
8: $Y = \{wsd_{ij} : sd_{ij} \in S_Q\}$
9: $PW = \frac{\sum_Y y}{|S_Q|}$
10: $P = \frac{CW + PW}{2}$
11: new_rank(c_j)=(1-β)*original_rank(c_j)+β*P
12: Rank-Documents(c_j, H, Q)
13: **end for**
14: OrderByRank(C)
15: **return** C

where β is the personalization factor. As in the work proposed in [23] β is used to decide the weight of the personalization in the computation of the rank varying from 0 to 1. We allow the user to choose the value of β in order to decide how much the desired results are to be near to her/his profile. If β is 0, the ranking is the same as plain search. If β is 1.0, then the search rank is totally determined by the profile. If β is 0.5, which is the default, the system considers equally the importance of the two contributions. In this algorithm CW (content weight) represents the weight of the semantic domains in the cluster which are also contained in the user profile w.r.t. the query. PW (profile weight) represents the weight given to the semantic domains w.r.t. to the terms of the query in the user profile.

7. PROFILE BASED QUERY DISAMBIGUATION

In addition to the query expanded using terms derived from clusters, we want to offer a set of user profile based disambiguated queries. In particular we show a set of terms taken from the semantic domains stored inside the user personal profile, highlighting them w.r.t. their frequency values stored in the User Profile matrix. The user can select one or more of these terms in order to build a new query that is submitted to the search engine. The main difference between the query built considering terms taken from clusters and the queries built using terms taken from the profile is that the first queries propose new original contents that are not correlated with the user's preferences while the others are closer to what the user usually searches while the others.

Algorithm 2 Rank-Documents (R, c, Q)

1: **for all** documents $d_k \in c$ **do**
2: for all qt consider the set ROW of rows of matrix R corresponding to the query terms
3: build the set COL of columns having only non zero values in the cells in the rows in R
4: insert in WS_Q the terms corresponding to COL
5: ws_k=web_source(d_k)
6: **if** $ws_k \in WS_Q$ **then**
7: assign to f the value in R corresponding to ws_k and qt
8: $WP = \frac{f}{MAX\_R}$
9: new_rank(d_k) = (1-β)*original_rank(d_k) + β*WP
10: **end if**
11: **end for**
12: OrderByRank(c)

8. CONCLUSIONS

In this paper we proposed a novel personalization technique based on the extraction of user-preferences information from the clustering of the user web searches results. The basic idea of the proposed approach is to store for each user information about the preferred semantic domains Web sources considered more reliable. The collected information are used to build a user profile useful to re-rank the results in order to offer the higher positions the results with a higher degree of semantic correlation with the user profile and originating from the more reliable Web sources.

9. REFERENCES

[1] G. Bordogna, A. Campi, G. Psaila, and S. Ronchi. A language for manipulating clustered web documents results. In *CIKM '08*, pages 23–32, New York, NY, USA, 2008. ACM.

[2] C.-H. Chang and C.-C. Hsu. Integrating query expansion and conceptual relevance feedback for personalized web information retrieval. *Comput. Netw. ISDN Syst.*, 30(1-7):621–623, 1998.

[3] P.-A. Chirita, C. S. Firan, and W. Nejdl. Summarizing local context to personalize global web search. In *CIKM '06*, pages 287–296, New York, NY, USA, 2006. ACM.

[4] P. A. Chirita, C. S. Firan, and W. Nejdl. Personalized query expansion for the web. In *SIGIR '07*, pages 7–14, New York, NY, USA, 2007. ACM.

[5] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *WWW '02*, pages 325–332, New York, NY, USA, 2002. ACM.

[6] E. N. Efthimiadis. User choices: a new yardstick for the evaluation of ranking algorithms for interactive query expansion. *Inf. Process. Manage.*, 31(4):605–620, 1995.

[7] Fellbaum. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press, May 1998.

[8] J. Gemmell, A. Shepitsen, M. Mobasher, and R. Burke. Personalization in folksonomies based on tag clustering. In *Proc. of the 6th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems*, July 2008.

[9] T. H. Haveliwala. Topic-sensitive pagerank. In *WWW '02*, pages 517–526, New York, NY, USA, 2002. ACM.

[10] R. Hemayati, W. Meng, and C. T. Yu. Semantic-based grouping of search engine results using wordnet. In *APWeb/WAIM*, pages 678–686, 2007.

[11] G. Jeh and J. Widom. Scaling personalized web search. In *WWW '03*, pages 271–279, New York, NY, USA, 2003. ACM.

[12] S.-B. Kim, H.-C. Seo, and H.-C. Rim. Information retrieval using word senses: root sense tagging approach. In *SIGIR '04*, pages 258–265, New York, NY, USA, 2004. ACM.

[13] R. Kraft and J. Zien. Mining anchor text for query refinement. In *WWW '04*, pages 666–674, New York, NY, USA, 2004. ACM.

[14] A. M. Lam-Adesina and G. J. F. Jones. Applying summarization techniques for term selection in relevance feedback. In *SIGIR '01*, pages 1–9, New York, NY, USA, 2001. ACM.

[15] F. Liu, C. Yu, and W. Meng. Personalized web search by mapping user queries to categories. In *CIKM '02*, 2002.

[16] S. Liu, F. Liu, C. Yu, and W. Meng. An effective approach to document retrieval via utilizing wordnet and recognizing phrases. In *SIGIR '04*, pages 266–272, New York, NY, USA, 2004. ACM.

[17] L. Nie, B. D. Davison, and X. Qi. Topical link analysis for web search. In *SIGIR '06*, pages 91–98, New York, NY, USA, 2006. ACM.

[18] S. Osinski, J. Stefanowski, and D. Weiss. Lingo: Search results clustering algorithm based on singular value decomposition. In *Intelligent Information Systems*, pages 359–368, 2004.

[19] F. Qiu and J. Cho. Automatic identification of user interest for personalized search. In *WWW '06*, pages 727–736, New York, NY, USA, 2006. ACM.

[20] C. Shah and W. B. Croft. Evaluating high accuracy retrieval techniques. In *SIGIR '04*, pages 2–9, New York, NY, USA, 2004. ACM.

[21] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *WWW '04*, 2004.

[22] J. Teevan, S. T. Dumais, and E. Horvitz. Characterizing the value of personalizing search. In *SIGIR '07*, pages 757–758, New York, NY, USA, 2007. ACM.

[23] J. wook Ahn, P. Brusilovsky, D. He, J. Grady, and Q. Li. Personalized web exploration with task models. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 1–10, New York, NY, USA, 2008. ACM.

[24] X. Wu, L. Zhang, and Y. Yu. Exploring social annotations for the semantic web. In *WWW '06*, pages 417–426, New York, NY, USA, 2006. ACM.

[25] D. Zhu and H. Dreher. Improving web search by categorization, clustering, and personalization. In *ADMA '08*, pages 659–666, Berlin, Heidelberg, 2008. Springer-Verlag.

Guiding Personal Choices in a Quality Contracts Driven Query Economy*

Huiming Qu
IBM Watson Research Center
hqu@us.ibm.com

Jie Xu
University of Pittsburgh
xujie@cs.pitt.edu

Alexandros Labrinidis
University of Pittsburgh
labrinid@cs.pitt.edu

ABSTRACT

The emergence of Web 2.0 has brought upon a plethora of database-driven web applications and services where both Quality of Service (QoS) and Quality of Data (QoD) are of paramount importance to end users. In our previous work, we have proposed Quality Contracts, a comprehensive framework for specifying multiple dimensions of QoS/QoD; we have also developed algorithms to maximize overall system performance under Quality Contracts. In this work, we turn our attention to the user side of the equation, on how to choose and adapt Quality Contracts to better serve users' needs in the presence of other users, who are competing for the same resources, in a virtual "economy" of Quality Contracts at the server. Towards this, we propose the Adaptive Quality Contract (AQC) scheme to maximize the success ratio of user queries. AQC switches between its Overbid (aggressive) mode and Deposit (conservative) mode, to allow users to survive through economic downturns and upturns. Extensive experiments with real traces show that our proposed scheme outperforms other competing schemes, under a variety of environments and a spectrum of workloads.

1. INTRODUCTION

How many times did you have to wait for your query to finish executing when visiting a travel reservation web site like Orbitz and Expedia? After getting the query results, how many times was the quoted price proved to be inaccurate when you clicked "buy this ticket"? This is just one example of a web-database system that illustrates the trade-off between Quality of Service (QoS) and Quality of Data (QoD). Clearly, some users would prefer fast response time, while tolerating slightly stale results (e.g., when they just want to find out about flight schedules). However, other users would instead prefer to get the most accurate query results, even if the response time was a bit higher (e.g., when they are ready to purchase a ticket). There are a number of issues that need to be addressed to implement a web-database system that is "receptive" to user preferences on QoS and QoD. We enumerate these next.

*Work supported in part by NSF Career award IIS-0746696 and NSF ITR award ANI-0325353.

PersDB '09, August 28, 2009, Lyon, France

Q1: How to describe user preferences? First and foremost, there needs to be a way for users to specify their preferences on QoS and QoD. One simple way would be to effectively assign users to equivalence classes (i.e., prefers QoS over QoD, or prefers QoD over QoS) and allow users to select which class they belong to. In our previous work, we have proposed (and advocated using) a more sophisticated framework, called *Quality Contracts* (QCs) [6] which is based on the micro-economic paradigm. The QC framework allows users to specify their preferences across a variety of quality dimensions. Similar proposals exist for other domains, such as the utility functions in real-time systems [15] and the service level agreements (SLAs) in Grid computing [3][1].

Q2: How are user preferences "implemented" to influence system decisions? Given a framework for users to specify their preferences over different quality dimensions, it is crucial to have a way to influence resource allocation decisions to maximize user satisfaction (i.e., compliance to user preferences). Towards this, we have developed admission control policies [13] and query & update scheduling algorithms [12] that maximize the overall system profit (to be gained by the server from satisfying QCs) and thus maximize the overall user satisfaction. The proposed algorithms are especially useful during periods of high server load, since they provide graceful service degradation.

In this work, we address another important problem that materializes after satisfactory solutions for questions Q1 and Q2 have been provided.

Q3: How should users adapt their Quality Contracts in the presence of competition? User preferences, if expressed through the Quality Contracts framework, include a constraint component (e.g., maximum acceptable data staleness) and a "worth" component (i.e., virtual money) for every quality dimension of interest to the user. In such an environment, always truthfully exposing the "worth" of the queries will not allow users to react to high competition (i.e., by "paying" a bit more than expected) nor to take advantage of reduced competition (i.e., by "paying" less than expected). In general, we consider the Quality Contracts submitted by the different users (along with their queries) as a *competitive economy*. As such, it is crucial for users to be able to adapt these QCs over time (while trying to achieve query quality that meets their preferences). In this paper, we propose user strategies to adapt QCs over time, during economic downturns (i.e., when the competition is less) and also during economic upturns (i.e., when the competition is higher).

Contributions The main contributions of this paper are:

- Given an environment where user preferences over different quality dimensions are expressed using Quality Contracts (QCs)

[1]We refer the reader to [6] for a detailed description of the QC framework and comparison to other approaches.

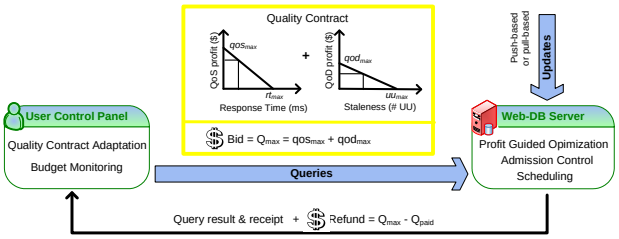


Figure 1: System Architecture

and are attached to queries, we look at the user viewpoint and, in particular, the framework for adapting QCs over time and the connection between a user's true preferences and his or her "exposed" QCs.

- We propose the *Adaptive Quality Contract (AQC)* strategy, which monitors a user's queries and the server's responses and automatically adapts the QCs of subsequent user-submitted queries. AQC[2] switches between two modes: *Overbid mode*, used at times of fierce competition among users; and *Deposit mode*, used at times of little competition.

We have demonstrated a QC-enabled web-database system during SIGMOD 2007 [14]. Our demo illustrated both the server view (for which the technical details were published in [12]) and a preliminary version of the user view, which is presented in this paper. In addition to introducing the AQC strategy (not in [14]), this paper explains its mathematical foundations, and presents a detailed experimental study using real traces.

2. SYSTEM ARCHITECTURE

We assume a web-database server architecture like the one in Figure 1. The system consists of two parts: the user module and the web-database server. Before describing these two parts, we discuss the basic concepts behind the QC economy.

2.1 The Quality Contract (QC) Economy

Economic mechanisms can be broadly classified into two types: commodity markets and auction markets [2]. Previous work has attempted to solve the system resource allocation problem under both paradigms. Under the commodity markets paradigm, commodities are exchanged in standardized contracts. Servers (acting as sellers) need to valuate their resources and assign prices for each unit. Users (acting as buyers) then decide from whom they buy the service to fulfill their queries. The shortcoming of commodity markets is the complexity and high cost for a server to valuate its resources, especially when the server workload fluctuates rapidly over time. To avoid this overhead at the server and make the valuation more precise, many systems follow the auction markets paradigm [2, 9, 16, 8, 19], where users need to give a price and bid on the resources or services provided by the server. Obviously, the uncertainty and burden of valuation never disappear; they are simply shifted to the user side. In this work, we adopt the auction markets paradigm. However, as we will elaborate in Section 3, we propose an adaptive bidding mechanism so that neither servers nor users have to worry about exact valuation.

In our system, users are allocated virtual money, which they spend in order to execute their queries according to their preferences; user preferences are described via QCs attached to each submitted query. Servers, on the other hand, execute users' queries and get virtual money in return for their service.

The virtual money is "paid" upon submission of a query to the server as part of the bidding (i.e., Q_{max}); any refund is given back along with the query results (i.e., $Q_{max} - Q_{paid}$). In our work, we follow a hedonic price model [17]; goods (i.e., services in our case) are priced by the users' valuation of different characteristics (QoS and QoD in our framework) and their contribution to users' utility. Towards this, we adopt the Quality Contracts (QC) framework as shown in Figure 1 for service pricing (by the users). A QC consists of a *QoS function* (where response time is mapped to QoS profit for the server) and a *QoD function* (where staleness is mapped to QoD profit for the server). The QoS and QoD metrics are application-dependent and orthogonal to our work.

Although our framework allows for users to specify complicated functions for QoS and QoD, in reality we expect users to simply select from a set of predefined such functions, much like most of our other digital "products" with different levels of service (e.g., cell phone plans).

In the presence of QCs, users and servers have distinct objectives: servers try to maximize their income, whereas users try to "stretch" their budget to execute as many queries as they can.

2.2 Server View

The web-database server is responsible for processing both updates and queries in order to meet the service requirements specified in the QC of each query.

Server Objective: Maximize Profit. The server objective is to maximize its profit, gained from each QC, through admission control and scheduling.

Server Optimization Mechanism: There are two phases of server optimization schemes: (1) admission control upon arrival of a query or an update, and (2) transaction scheduling once admitted. In general, the higher the bid, the higher the chance that a query is admitted and completed with high quality. Due to the space limitation, please refer to [13, 12] for more details on these two phases. We adopted Two Phase Locking - High Priority (2PL-HP) [1] where the lower priority transaction releases the lock to the higher priority transaction at a conflict.

2.3 User View

The user aspect of the system must include an interface for users to specify QCs and the ability to monitor the execution of QC-enabled queries, while keeping track of the current budget. Although the QC framework empowers users to influence resource allocation decisions at the server (to better meet their preferences), it also places the burden on the users to choose QCs (and adapt them over time). We expect that users will employ *user agents*, which will have explicit "instructions" from each user (on his/her true preferences and budget constraints) and a QC adaptation strategy.

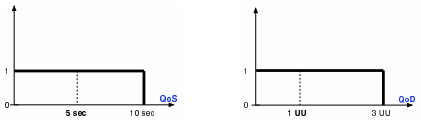
Quality Contract / User Satisfaction: In this paper, we adopt QCs with linearly decreasing positive functions [12]. Intuitively, users can set the following four parameters to define a QC:

- qos_{max}, the maximum QoS profit,
- qod_{max}, the maximum QoD profit,
- rt_{max}, the maximum bearable response time, and
- uu_{max}, the maximum bearable staleness.

In this work, we simplify our model with an equivalent presentation, where the first two parameters (qos_{max} and qod_{max}) are replaced by:

[2]AQC is pronounced AQuaC, which sounds like AFLAC; however, we do not have a fancy mascot.

- Q_{max}: qos_{max} +qod_{max}, the maximum payment for the query.
- γ: $\frac{qodmax}{qosmax}$, relative importance between QoS and QoD.



(a) QoS (b) QoD

Figure 2: User Satisfaction Functions Example

As discussed in the introduction section (question Q3), in specifying QCs, users may not want to reveal the true worth of their queries (e.g., how much they can pay for a query result with a certain response time), although they would be willing to reveal their constraints (i.e., would prefer an answer within 10 seconds). In other words, the constraints in QCs are truthful, but the mapping to the worth dimention may not be. Figure 2 shows a simple example of what "truthful" *user satisfaction functions* might look like. Under this setting, queries are considered acceptable as long as the results meet the constraints on both QoS and QoD. Without loss of generality, we adopt such binary-step user satisfaction functions in this paper.

Following the example of Figure 2, we define two outcomes for a query for the general case:

- **Success:** A query succeeds if it is returned with valuable answers, meaning that the response time is shorter than the QoS constraint, rt_{max}, and the staleness is smaller than the QoD constraint, uu_{max}. Successful queries give to the server a nonzero payment, $Q_{paid} > 0$. The actual value of Q_{paid} depends on how well the server executes the query, given the QC. In terms of the user satisfaction functions, a successful query yields 1 from the product of all user satisfaction functions of the query (i.e., across all quality dimensions).
- **Failure:** If a query fails either the QoS or the QoD constraint, we call the query a *failure*, and Q_{paid} = 0. This allows the user to also infer the data freshness of the returned results.

User Objective: Success Ratio Maximization. The users' goal is to adapt Quality Contracts (e.g., by changing Q_{max}) to get as many as possible of his/her queries executed successfully, within the given total budget.

2.4 Analysis of Existing QC Adaptation Schemes

Assuming a user with N queries to submit and a total budget B, we consider three baseline strategies, which compute $Q^{(i)}_{max}$, the total bid for the QC of query i, as follows:

- **Fixed (FIX):** $Q^{(i)}_{max} = \frac{B}{N}$. FIX is a static policy, which assigns each query an equal share of the total budget.
- **Random (RAN):** $Q^{(i)}_{max}$ =*uniform*$[\frac{B}{N} - c,\ \frac{B}{N} + c]$, where c is a constant. This strategy uses $\frac{B}{N}$ as the mean, and $[\frac{B}{N} - c,\ \frac{B}{N} + c]$ as the range to pick Q_{max} uniformly.
- **Dynamic (DYN):** $Q^{(i)}_{max} = \frac{B_i}{N-i}$. This scheme monitors the current budget left B_i and the number of queries left $N - i$ before query i is issued.

Problems with existing schemes: FIX does not make full use of the budget, because it ignores the refunds from the previous failed queries. The RAN scheme is similarly problematic. DYN addresses the issue of ignored refunds by dynamically updating the available budget. However, DYN favors the queries issued later than earlier and creates an unfair allocation of the total budget.

3. ADAPTIVE QUALITY CONTRACT (AQC)

In this section, we present our proposed *Adaptive Quality Contract Scheme (AQC)*, which addresses the problems and limitations of the baseline algorithms that were presented in the previous section. Our AQC scheme switches between two modes: *Overbid mode* (Section 3.1) and *Deposit mode* (Section 3.2); we discuss how AQC chooses between the two modes in Section 3.3.

3.1 Overbid Mode

As we have shown, DYN unfairly "favors" later queries by monotonically increasing Q_{max} as time progresses using the cumulative refunds from previously finished queries. This behavior is roughly equivalent to last-minute spending by companies at the end of a fiscal year, since at that time, any of the remaining money in the current year's budget will effectively disappear unless spent immediately.

The Overbid mode of AQC addresses this problem by setting the budget of the submitted quality contracts for each query to be such that the *expected payments sum up to the overall budget*. In contrast, the DYN scheme sets the bid per query to be such that the individual bids sum up to the total budget (which clearly results in under-utilization of the budget, until the last minute).

In order to make the expected payments sum up to the overall budget, we need to make sure that the expected payments for the i^{th} query sum up to its fair share of the budget:

$$\mathbf{E}_p[Q^{(i)}_{paid}(x,\ y)] = \text{budget per query} = \frac{B_i}{N-i} \tag{1}$$

Then, in order to find how to set the QC for the query, we have to essentially express Q_{paid} in terms of Q_{max}, and solve Equation 1 for Q_{max}. Q_{paid} depends on the QoS function S, QoD function D, and how well the server returns the query (response time x and staleness y). Thus, as we show next, the expectation of Q_{paid} over the probability distribution of response time (x) and staleness (y) can be expanded as the sum of expected expenditure from the QoS function and from the QoD function respectively:

$$\mathbf{E}_p[Q^{(i)}_{paid}(x,\ y)] = \mathbf{E}_p[S(x)] + \mathbf{E}_p[D(y)] \tag{2}$$

If we combine Equation 1 with Equation 2, we have that:

$$\mathbf{E}_p[S(x)] + \mathbf{E}_p[D(y)] = \frac{B_i}{N-i} \tag{3}$$

In this work, we adopt linear segmented QCs where the QoS function can be represented as in Equation 4 and the QoD function can be represented as in Equation 5. If other formats of QC functions are adopted, Equation 4 and Equation 5 should be modified accordingly.

$$S(x) = \begin{cases} qos_{max}(1 - \frac{x}{rt_{max}}) & \text{if } x \in [0,\ rt_{max}] \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$D(y) = \begin{cases} qod_{max}(1 - \frac{y}{uu_{max}}) & \text{if } y \in [0,\ uu_{max}] \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

We compute the expectation of QoS profit using empirical expectation, as shown in Equation 6.

$$\begin{aligned} \mathbf{E}_p[S(x)] &= \int S(x)p(x)\,dx \\ &= qos_{max}\int_0^{rt_{max}} p(x)\,dx - \frac{qos_{max}}{rt_{max}}\int_0^{rt_{max}} xp(x)\,dx \\ &\approx qos_{max}(\mathrm{P}(x < rt_{max}) - \frac{\bar{x}}{rt_{max}}) \end{aligned} \tag{6}$$

where $\mathrm{P}(x < rt_{max})$ is the percentage of cases that the response time of the user query is smaller than its response time constraint rt_{max}, and $\bar{x}$ is the average response time. Both $\mathrm{P}(x < rt_{max})$ and $\bar{x}$ can be computed based on the query execution history. We introduce α to denote this part of computation and summarize the expected QoS profit as follows:

$$\mathbf{E}_p[S(x)] \approx qos_{max} \cdot \alpha \tag{7}$$
$$\alpha = \mathrm{P}(x < rt_{max}) - \frac{\bar{x}}{rt_{max}}$$

Similarly, we compute the expectation of QoD profit:

$$\mathbf{E}_p[D(y)] \approx qod_{max} \cdot \beta \tag{8}$$
$$\beta = \mathrm{P}(y < uu_{max}) - \frac{\bar{y}}{uu_{max}}$$

As described in Equation 3, the total expected profit from both QoS (Equation 7) and QoD (Equation 8) should be set to the current budget per query $\frac{B_i}{N-i}$:

$$qos_{max} \cdot \alpha + qod_{max} \cdot \beta = \frac{B_i}{N-i} \tag{9}$$

where α and β are computed based on query execution history (as shown in Equation 7 and Equation 8). Since the ratio between qos_{max} and qod_{max} is known as γ, we have:

$$Q_{max} = qos_{max} + qod_{max}$$
$$qod_{max} = \gamma \cdot qos_{max} \tag{10}$$

We solve Equations 9 and 10 to get the final solution of Q_{max}:

$$Q_{max}^{(i)} = \frac{B_i}{N-i} \cdot \frac{1}{\alpha + \gamma \cdot \beta} \tag{11}$$

In the above solution, $\frac{1}{\alpha+\gamma\cdot\beta}$ is essentially the overbid factor.

3.2 Deposit Mode

Although Overbid mode successfully utilizes as much of the budget as possible (and in a fair manner across all queries), it will not detect the cases of "overpayment" because of the server having a light load. In such cases, there is not much "competition" from other users, and as such the user could have paid less than what Overbid mode would suggest.

The benefit of detecting these cases comes from the inherent dynamic nature of typical web-database servers. The load at such servers can fluctuate from *very high* (e.g., in periods of flash crowds), where queries would require a high budget or they will not be able to execute, to relatively *low*, where queries would require a much less budget than usual to execute.

In order to make sure that the AQC scheme can successfully react to the inherent dynamic nature of web-database servers, we introduce a budget saving scheme which we call *Deposit mode*. The main idea behind Deposit mode is to recognize cases when users can spend less of their budget (because of a less competitive situation), so that they are ready to spend more when facing stronger competition from other users.

To implement Deposit mode, the Q_{max} is reduced when there is a row of consecutive successful query executions. Let $Q_{max}^{(s)}$, $Q_{paid}^{(s)}$ be the budget and the payment for the most recent successful query. We set the new $Q_{max}^{(i)}$ in Deposit mode as follows:

$$Q_{max}^{(i)} = Q_{max}^{(s)} \cdot (1 - \frac{Q_{paid}^{(s)}}{Q_{max}^{(s)}}) \tag{12}$$

We call the ratio of $\frac{Q_{paid}^{(s)}}{Q_{max}^{(s)}}$ as the *deposit factor*. Notice that the closer $Q_{paid}^{(s)}$ is to $Q_{max}^{(s)}$, the bigger the deposit factor is. The intuition is that we could deposit more and bid less when historical success comes with very good performance (a high $Q_{paid}^{(s)}$ usually corresponds to high QoS and high QoD). A high deposit factor thus may indicate that the system is currently lightly loaded. Although a lower bid will decrease the priority of the query, hopefully in a lightly loaded server, the query can still be answered within constraints. On the contrary, if the last successful query barely meets the QoS and QoD constraints, the deposit factor will be close to zero and the query will be kept with a competitive bid.

3.3 Switching between Deposit and Overbid

At the beginning, the system is set to the overbid mode by default. AQC keeps track of the number of consecutive query successes ($successQ.size$) and uses it to decide the current system mode.

If the number of successes is significantly large (i.e., larger than a threshold c), the system is set to deposit mode. This is because a consecutive successful query history indicates a less competitive environment or a lightly loaded web-database server, thus the bid can potentially be decreased without hurting the success ratio.

Notice that $successQ.size$ only includes those queries that are completed within the time window w. Thus, $successQ.size$ may decrease due to two reasons: (1) there are no more queries to be completed (i.e., neither query success nor query failure), as a result, $successQ.size$ decreases as the moving window w moves on. If $successQ.size$ drops below c, the system mode will be set to overbid because of the lack of successful feedbacks; (2) there is a query failure, which will reset $successQ.size$ to zero immediately. In both cases, system mode is set to overbid promptly to utilize the user's budget as much as possible so that the server is motivated to execute the users queries with higher priorities.

By switching between the overbid and deposit modes according to the query success/failure, the AQC scheme naturally follows the law of supply and demand. We expect the overhead of adaptation to be linear to the total number of queries processed.

4. EXPERIMENTAL RESULTS

4.1 Experimental Setup

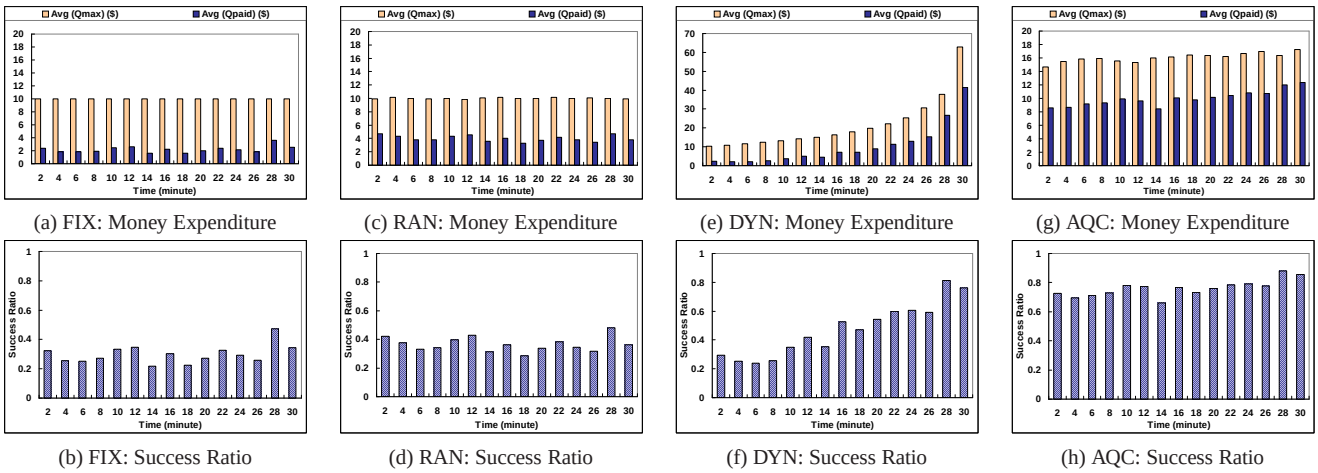
We have acquired access traces from a popular stock market information web site, Quote.com, and combined them with the NYSE (New York Stock Exchange) update traces for the same time period, which enabled us to accurately generate both query and update workloads for our experiments.

Query Traces We use real queries from Quote.com for April 24, 2000. All queries are read-only. We concentrated on a "heavy" workload for the server, a 30-minute (10:30am-11:00am) interval with over 120,000 queries on 4,107 different stock symbols.

Update Traces We extracted the actual trades on all securities listed on NYSE during the same time interval as our query trace (10:30am-11:00am on April 24, 2000). The update trace shares the same indexing scheme with the query trace. The update trace fragment we used has over 396,000 entries.

Comparison Algorithms To evaluate our proposed QC adaptation strategy, we performed an extensive simulation study using the FIX, RAN, DYN schemes (Section 2.4) and our proposed proposed AQC strategy (Section 3).

Each query is submitted to the system along with a user-specified QC; each user also has an initial budget, which for simplicity is



(a) FIX: Money Expenditure (c) RAN: Money Expenditure (e) DYN: Money Expenditure (g) AQC: Money Expenditure

(b) FIX: Success Ratio (d) RAN: Success Ratio (f) DYN: Success Ratio (h) AQC: Success Ratio

Figure 3: Duet Over time

equal among all users.

4.2 Performance Comparison

For a fair comparison, we present results from two different execution runs: duet and quartet. In duet, each run contains two classes of users (i.e. two algorithms), creating a one-to-one confrontation to show directly which algorithm performs better. In quartet, we put all four algorithms into the run, where they all interact and compete with each other.

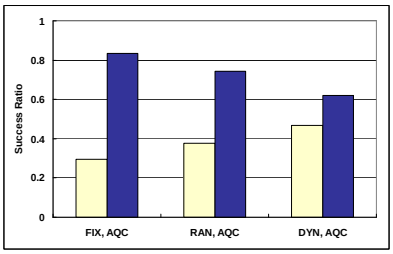


Figure 4: Duet Environment: AQC vs. Baseline Algorithms. AQC outperforms baseline algorithms by up to 183%.

4.2.1 Duet

Experiment Design (Figure 4) We compare AQC with each baseline algorithm individually to eliminate unnecessary interactions from multiple algorithms. We performed three runs: (FIX, AQC), (RAN, AQC), and (DYN, AQC). Each query from the trace is randomly associated with one of the two algorithms in the experiment.
Results (Figure 4) We run each trace 20 times and report the average query success ratio for the three comparisons in Figure 4. The performance difference is quite obvious: AQC users perform 183% better than FIX, almost 100% better than RAN, and more than 30% better than DYN.
Results Over Time(Figure 3) Given that the four algorithms have different behavior over time, we also plot the bid Q_{max} and the money paid Q_{paid} for each query as well as the query success ratio with a 2-minute window over time.

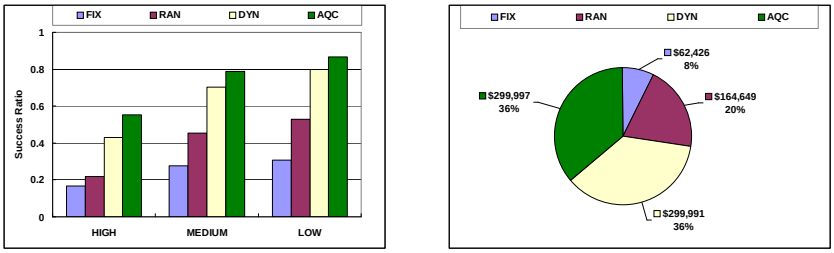
In Figure 3(a), FIX gives a constant bid ($10) for each query. Due to the unavoidable CPU time and unpredictable queuing time, the real expenditure is only around $2 on average. Failing to reuse the refund leaves FIX with a small success ratio shown in Figure 3(b). RAN has similar results as shown in Figure 3(c) and (d). Q_{max} varies around $10 and Q_{paid} is around $4 on average. With more than half the budget wasted, RAN gains less than 50% success ratio over time.

In Figure 3(e), we see that DYN dynamically adjusts the current available budget and increases Q_{max} over time. As a result, DYN's success ratio increases over time too, as shown in Figure 3(f). However, DYN is still conservative on early issued queries, which not only jeopardizes the fairness of queries coming at different times, but also hurts its overall success ratio.

Finally, Figure 3(g) and (h) show AQC's improvement from two sides. First, the average Q_{paid} is around $10, thus the budget is fully used to increase the quality of query results. AQC is able to set Q_{max} higher than $10 because of foreseeing the expected expenditure. Second, AQC tries to save money after consecutive successes, so that it can bid higher to survive a tougher situation later. This is why we see a few downward trends in Figure 3(g). Both aspects help AQC achieve significantly better results when it competes with other algorithms.

4.2.2 Quartet

Experiment Design (Figure 5) Having compared the different baseline algorithms (FIX, RAN, DYN) to our proposed algorithm (AQC), we mix the four user algorithms in the same execution run; each class of users has 30,000 queries with a mean Q_{max} of $10. In this set of experiments, we focus on (1) varying quality constraints (Figure 5(a)), and (2) showing both the user view and the server view (Figure 5(b)).
Results (Figure 5(a)) We change the user constraints on QoS to be tight, medium, and loose to generate three traces, High, Medium, and Low workload respectively. As expected, for all algorithms, the success ratio is higher with Low system workload (which has loose quality constraints). In comparison to other algorithms, AQC performs the best under the whole spectrum of workloads. Another observation is that a high system workload also exaggerates the performance differences among the algorithms. Under high workload, AQC outperforms FIX by 233%. AQC also achieves 155% better performance than RAN and 28% better than DYN.
Results (Figure 5(b)) In addition to the users' view of these algorithms, we also show the server profit gains from each user algorithm under the medium workload. We observed similar trends with both the high and the low workload. Figure 5(b) shows that the server stands to gain much more profit from DYN and AQC, thus tends to serve them better than FIX and RAN. Making full use



(a) User View: Success Ratio (b) Server View: Profit

Figure 5: Quartet Environment: 4 algorithms under different workload settings.

of the budget is a win-win strategy from both users' and server's point of view.

To summarize, AQC not only gives the best success ratio under various workloads, but also makes the users most popular from a system's point of view, as the system can make much more profit from users utilizing AQC.

5. RELATED WORK

Web-databases There is a plethora of papers that focus on improving the performance of user requests to database-driven web sites, using caching [4, 11] or materialization [7]. These approaches usually provide a best-effort solution in terms of either QoS or QoD. In our recent work [6], we introduced the Quality Contract framework to combine individual users' preferences for both QoS and QoD. We demonstrated the QC framework in [14], in combination with our policies for admission control [13], and query & update scheduling [12]. Our demo also highlighted the benefits of user-side adaptation of QCs (although we did not provide the AQC scheme, as we do in this paper).

Grids and Web Services Service Level Agreements (SLAs) in Grid applications [3] is another related area. In SLAs, resource availability, capability, and cost are considered for effective resource management. SLAs also exist for Web-services [10, 18, 20]. Although sharing the general goal of resource regulation and cost controlling, our work focuses on one specific resources: web-databases.

6. CONCLUSIONS

In previous work we have proposed the Quality Contracts (QCs) framework, and introduced the supporting system optimizations. In this work, we turn our attention to the user side of the equation and consider *user strategies to adapt Quality Contracts over time*. Towards this, we proposed the Adaptive Quality Contract (AQC) strategy, which monitors a user's queries and the server's responses in order to automatically adapt the QCs of subsequent user-submitted queries. We performed an extensive simulation study with real traces, which showed that AQC consistently outperforms baseline algorithms (by up to 233%).

Currently, we are exploring strategy-proof mechanisms for the bidding process and are considering how other schedulers (e.g., [5]) could be adapted to "understand" Quality Contracts.

7. REFERENCES

[1] R. K. Abbott and H. Garcia-Molina. Scheduling real-time transactions: A performance evaluation. *ACM Transactions on Database Systems*, 17(3):513–560, 1992.

[2] A. AuYoung, B. Chun, A. Snoeren, and A. Vahdat. Resource allocation in federated distributed computing infrastructures. In *Proc. of the 1st Workshop on Operating System and Architectural Support for the Ondemand IT InfraStructure*, Oct 2004.

[3] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. *Proceedings of the IEEE*, 93(3):698–714, 2005.

[4] A. Datta et al. Proxy-Based Acceleration of Dynamically Generated Content on the World Wide Web: An Approach and Implementation. In *SIGMOD Conference*, 2002.

[5] S. Guirguis, M. A. Sharaf, P. K. Chrysanthis, A. Labrinidis, and K. Pruhs. Adaptive scheduling of web transactions. In *ICDE Conference*, April 2009.

[6] A. Labrinidis, H. Qu, and J. Xu. Quality contracts for real-time enterprises. In *BIRTE Workshop*, 2006.

[7] A. Labrinidis and N. Roussopoulos. Webview materialization. In *SIGMOD Conference*, 2000.

[8] K. Lai, L. Rasmusson, E. Adar, L. Zhang, and B. A. Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent Grid Syst.*, 1(3):169–182, 2005.

[9] H. C. Lau, S. F. Cheng, T. Y. Leong, J. H. Park, and Z. Zhao. Multi-period combinatorial auction mechanism for distributed resource allocation and scheduling. In *IAT*, 2007.

[10] Y. Liu, A. H. Ngu, and L. Z. Zeng. Qos computation and policing in dynamic web service selection. In *WWW Alt.*, pages 66–73, 2004.

[11] Q. Luo, S. Krishnamurthy, C. Mohan, H. Pirahesh, H. Woo, B. G. Lindsay, and J. F. Naughton. Middle-tier Database Caching for e-Business. In *SIGMOD Conference*, 2002.

[12] H. Qu and A. Labrinidis. Preference-aware query and update scheduling in web-databases. In *ICDE Conference*, 2007.

[13] H. Qu, A. Labrinidis, and D. Mosse. Unit: User-centric transaction management in web-database systems. In *ICDE Conference*, 2006.

[14] H. Qu, J. Xu, and A. Labrinidis. Demo: Quality is in the eye of the beholder: Towards user-centric web-databases. In *SIGMOD Conference*, 2007.

[15] K. Ramamritham and J. Stankovic. Scheduling algorithms and operating systems support for real-time systems. *Proceedings of the IEEE*, 82(1):55–67, 1994.

[16] O. Regev and N. Nisan. The popcorn market—an online market for computational resources. In *Proc. of the first international conference on Information and computation economies*, 1998.

[17] S. Rosen. Hedonic prices and implicit markets: Product differentiation in pure competition. *The Journal of Political Economy*, 82(1):34–55, Janurary - February 1974.

[18] A. Sahai, V. Machiraju, M. Sayal, L. J. Jin, and F. Casati. Automated sla monitoring for web services. In *IEEE/IFIP DSOM*, 2002.

[19] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A distributed computational economy. *IEEE Trans. on Software Engineering*, 18(2):103–117, February 1992.

[20] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Z. Sheng. Quality driven web services composition. In *WWW Conference*, 2003.

Context-Aware Recommender Systems: A Service-Oriented Approach

Sofiane Abbar
PRiSM Lab, Versailles university
45 avenue des Etats-Unis
78035, Versailles, France
sofa@prism.uvsq.fr

Mokrane Bouzeghoub
PRiSM Laboratory, Versailles university
45 avenue des Etats-Unis
78035, Versailles, France
mok@prism.uvsq.fr

Stéphane Lopez
PRiSM Lab, Versailles university
45 avenue des Etats-Unis
78035, Versailles, France
hal@prism.uvsq.fr

ABSTRACT

Recommender systems are efficient tools that overcome the information overload problem by providing users with the most relevant contents. This is generally done through user's preferences/ratings acquired from log files of his former sessions. Besides these preferences, taking into account the interaction context of the user will improve the relevancy of recommendation process. In this paper, we propose a context-aware recommender system based on both user profile and context. The approach we present is based on a previous work on data personalization which leads to the definition of a Personalized Access Model that provides a set of personalization services. We show how these services can be deployed in order to provide advanced context-aware recommender systems.

1. INTRODUCTION

The last decade met a remarkable proliferation of P2P networks, PDMS, semantic web, communitarian websites, electronic stores, etc. resulting in an overload of available information. Current information systems deal with a huge amount of content, and deliver in consequence a high number of results in response to user queries. Thus, users are not able to distinguish relevant contents from secondary ones.

Recommender systems (RS) are efficient tools designed to overcome the information overload problem by providing users with the most relevant content [8]. Recommendations are computed by predicting user's ratings on some contents. Rating predictions are usually based on a user profiling model that summarizes former user's behaviour.

The importance of RS is now well established. Netflix organizes a contest [1] in which one million dollar is offered for any better recommendation engine. This contest shows, in one hand, the importance that industrials give to RS, and in another hand that better recommendations worth a lot. Two important questions arise from the Netflix contest: (i) What is a better RS? And (ii) How can a system provide the best recommendations?

In our vision, a better RS is the one which delivers recommendations that best match with users' preferences, needs and hopes at the right moment, in the right place and on the right media. This can't be achieved without designing a RS that takes into account all information and parameters that influence user's ratings. These information may concern demographic data, preferences about user's domain of interest, quality and delivery requirements as well as the time of the interaction, the location, the media, the cognitive status of the user and his availability. This knowledge is organized into the two concepts of user profile and context. The user profile groups information that characterizes the user himself while the context encompasses a set of features which describe the environment within which the user interacts with the system.

We claim that taking into account both profiles and contexts in a recommendation process benefits to any RS for many reasons: (*i.*) Users' preferences/ratings change according to their contexts [5, 12]. (*ii.*) The additive nature of traditional RS [6] does not consider multiple ratings of the same content. (*iii.*) RS may fail in providing some valuable recommendations as their similarity distance is uniformly applied to user preferences without analyzing the discrepancies introduced by the context.

In [2], we proposed a Personalized Access Model (PAM), a framework that provides a set of personalization concepts and services, generic enough to be applied to a large variety of applications. Based on the PAM, and following the efforts started by Adomavicius et al. [3, 4] and Anand et al. [6], we propose a Context-Aware Recommender System (CARS) which is based on both user profiles and contexts. In our approach the same user who interacts from different contexts is provided with different recommendations.

Our main contributions include the following: (*i.*) We present a general architecture for context-aware RS based on a set of personalization services. (*ii.*) We extend the PAM with a new context learning service that enables concrete construction of users' contexts from their log files. (*iii.*) The contextualization service proposed in [1] is improved by combining both *support* and *confidence* of ratings within a given context instead of considering their frequencies only. Notice that this paper reports a conceptualization effort for integrating context into RS. The evaluation of CARS approach is under study.

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

VLDB '09, August 24-28, 2009, Lyon, France

This paper is organized as follow: Section 2 presents a high level architecture of CARS and poses the main requirements that CARS should satisfy. Section 3 recalls concepts and services provided by the PAM. Sections 4 and 5 focus on the services applied to CARS. Section 6 provides a global view on CARS deployment using PAM services. Section 7 summarizes related works. Section 8 concludes the paper with further research.

2. REQUIREMENTS FOR CARS

We focus on RS which combine content-based techniques and Collabotrative Filtering (CF). The content-based approach permits to learn users' profiles by analyzing content descriptors. Resulting profiles are, then, matched and compared to determine similar users in order to make collaborative recommendations. The CF approach allows exploiting the ratings given by the Top K neighbors of the active user in order to derive the missing rating by aggregation function. Figure 1 gives a flavour of the CARS global architecture.

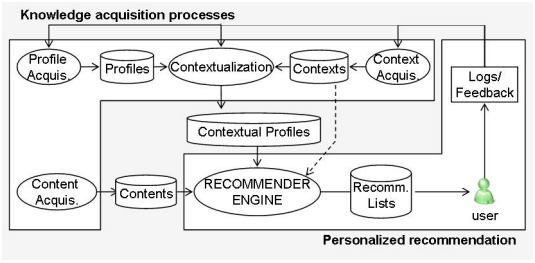


Figure 1: Global architecture of context-aware RS

This architecture is composed of: the left-upper block which concerns knowledge acquisition and the right-bottom block which concerns personalized recommendation.

Knowledge acquisition processes: This block is responsible for the acquisition and the management of knowledge a CARS needs to process recommendations. This knowledge is grouped into three entities: user profiles, content descriptors, and contexts. We focus in this paper on: (i) context acquisition from log files, and (ii) discovery of relationships between these contexts and user profiles elements (called contextual profiles).

Personalized recommendation: This block encompasses actual RS operations. The recommender engine (RE) takes as inputs (i) contextual profiles generated in the knowledge acquisition block and (ii) the current context of the target user to compute a list of contextual recommendations. The behaviour of the target user when consuming these recommendations is stored in log files on which acquisition processes are based to update profiles, contexts and their mappings; hence, closing the cycle between knowledge acquisition and their exploitation.

The design of such CARS is driven by the following requirements:

1. Distinguish profiles from contexts for a well-understood definition and evolution of both entities.
2. Enable contexts discovery through a concrete approach and a widely available data (standard log files).
3. Consider both profiles and contexts into recommendation such that a user interacting from different contexts would be provided with appropriate recommendations.

As a consequence of the separation of profile and context, applications may be only aware on profiles or contexts or on both. The next section details a service-oriented approach for designing CARS that satisfy above requirements.

3. OVERVIEW OF PAM CONCEPTS AND SERVICES

Our CARS proposal is based on a former work [2] in which we defined a Personalized Access Model (PAM) that provides a set of personalization services. The PAM aims at providing a generic set of concepts and services which can be deployed over a given architecture to make applications aware of users' profiles and contexts.

The components of the PAM are built around profile and context meta models which are adaptable to a wide range of applications and which are open to integrate specific knowledge not included initially. Profile and context meta models are organized into dimensions and sub dimensions which are described by (attribute, value) couples.

3.1 Main Concepts

This section provides an informal definition of the concepts that will be used in the following sections.

User profile. Provides an extensive definition of the preferences that a user has in a given domain of interest. It is a set of (attribute,value) couples rated by the user or the RS on the basis of user's actions.

Active User. In RS, an active user is the one for whom recommendations are calculated.

Context. Set of features characterizing the environment within which users interact with RS.

Active context. It is the context within which the active user interacts with RS. It is also called current context.

Contextualized profile. It's a set of mappings that relate a subset of profile preferences to the context in which they are defined.

Operational profile. It's a runtime binding of a contextualized profile to the current context of the user.

3.2 PAM Services

The PAM provides several services which have been progressively extended, following application requirements [2]. Among these services, we distinguish offline (Contextualization, Context discovery) and online services (Binding, Matching). The former set is used at the design time of a personalized application, while the latter is used at the execution time. We list hereafter those which are worth to be used in personalized RS.

Context discovery service. Discovers, for each user, regular contexts within which he interacts with RS (e.g. Holiday context, Office context).

Contextualization Service. Aims at identifying mappings between user ratings and contexts in which these ratings were defined. Mappings are of the form (R_i, c_j, s_{ij}), where $R_i = Item_i \times r_i$, and s_{ij} represents the importance of the rating r_i over $item_i$ within the context c_j.

Binding Service. Exploits the mappings generated by the contextualization service to derive the operational profile.

Matching Services Allow capturing the semantic similarity between two concepts. The similarity value takes into account the structure of compared concepts, user ratings and

ontologies with which items can be described. We distinguish four matching services [2]: profile/profile, context/context, item/item, and profile/item. Due to space limitation, matching services are not devellopped in this paper.

4. OFFLINE SERVICES

This section details the two services that acquire the necessary knowledge to efficiently run the CARS. These services are an answer to the first two requirements imposed in section 2. For each service, we give its definition, its deployment on CARS, and details on its operational semantics.

4.1 Context Discovery Service

4.1.1 Service definition

The goal of the context discovery service is to learn from user log files, the most recurrent contexts (situations) within which the user interacts in general.

$$C : Context\_Discovery(\alpha, L) \quad (1)$$

The context discovery service takes as input the user log file (L) tracked from former recommendations and returns a set of regular contexts C within which a user, whose profile is P_u, interacts with the personalized application.

4.1.2 Service deployment on CARS

The approach we propose to discover contexts is based on the analysis of logs to capture regular contexts. Hereafter, a motivating example is given to understand the approach.

Example. Assume that a user, John, interacts frequently with RS. Then, thanks to information contained in his log files, the system can capture that John's interests (activities) change according to the IP address, the date and time, the device, etc. This can be pointed by a lookup to his log file from where two contexts can be derived: office context during working days and home context during weekends.

John / IP 192.168.53.25 /May 4 17h:00 /winXP,En,Firefox5.0
John / IP 192.168.53.25 /May 5 17h:30 /winXP,En,Firefox5.0
John / IP 192.168.53.25 /May 4 17h:33 /winXP,En,Firefox5.0
...
John / IP 192.168.70.1 /May 9 12h:00 /MacOS X,En,Safari
John / IP 192.168.70.1 /May 9 20h:00 /MacOS X,En,Safari

Obviously, the content to be recommended change according to John's context (e.g. scientific papers for office context and movies for home context). Context discovery from logs assumes the manipulation of well structured log file. Hopefully, the analysis of the most known log file formats such as Apache CLF, IIS LF and W3C LF [11] reveals that most of them already contain contextual information (fields) that correspond to the context meta model proposed in [2]. Among these fields there are, for example, the IP Address, the DATE of the request, and the USER-AGENT (browser, OS, ...).

4.1.3 Log File Format

The log file format we used is based on the W3C log file format [11] which is enough extensible (thanks to ten W3C General Purpose Fields) to adapt it to various applications. field: $Date\ Time\ c-ip\ c-dns\ c-auth-id\ cs-method\ cs-uri-stem\ cs-uri-query\ cs(UserAgent)\ sessionId\ c-Action\ c-Device$. Notice that this log file format defines two new fields only which are $c-action$, and $c-device$. The first one is well known in application server log files, it contains usually the id of the action (e.g. Buy) that user applies to a given content. The second field informs about the device used (e.g. Laptop, a mobile phone, a remote control, etc.).

Id	IP	Date	Time	Agent	Device	Content	...
1	x	y	t	a	d	Item1	...
2	e	r	f	d	c	Item2	...
3	r	g	f	q	s	Item1	...

Figure 2: Log file

Among the log file format fields, five are contextual:

$Date$: the domain of date is organized into a hierarchy, in such a manner that we can know if it is a working day or a weekend, holiday or not, etc.

$Time$: a day is partitioned into periods which are organized into a hierarchy. The time can inform about the day period: day, night, morning, afternoon, etc.

$c-ip$: The IP address in used to localize the user, the localities are organized into a hierarchy of continent, region, country, city, town, etc.

$cs(UserAgent)$: gives information about the user Browser and OS (operating system) further than their languages.

$c-Device$: characterizes the used device for the user interaction, devices can be segmented into a hierarchy.

Conjunction of these attribute represent contexts within which users interacts with the RS. We propose to group these contexts into clusters, each representing a regular context or situation such as home, labs, etc.

4.1.4 Context Discovery Algorithm

Log files are transformed and uploaded into a database relation having the schema of figure 2. Based on this table, contexts are discovered by mining the five contextual attributes given above.

Let a context tuple be a conjunction of the five contextual attributes which corresponds to one row (see figure 2).

The idea is to cluster the set of all tuples into a finite set of clusters, each cluster representing a particular regular context. Thus, we start by applying the *Agglomerative Hierarchical Clustering (AHC)* [10] in order to estimate the number of clusters k, then, we apply the *k-means* algorithm with the k center of resulting clusters in AHC algorithm instead of choosing arbitrarily the initial k centers.

Algorithm 1 starts by extracting context instances from the user log file. In line 3, AHC is applied on these instances in order to compute the initial k clusters. Then, from lines 4 to 13, *k-means* is applied as an iterative relocation algorithm in order to improve the initial clustering obtained with AHC. The algorithm iterates until Squared Error $\mathcal{E}$ achieves its minimal value.

4.2 Contextualization Service

4.2.1 Service definition

The main idea of the contextualization is to check whether there are correlations between the user profile elements and the user feedback within a given context.

$$M(P_u, C) : Contextualization(P_u, H) \quad (2)$$

Contextualization process takes as input a user profile P_u and the user history H corresponding to the user feedback in contexts C discovered previously. It returns a set of contextual mappings M representing dependencies between el-

Algorithm 1 Automatic Contexts Discovery

Input: the user log file $\mathcal{L}$, the threshold α.
Output: the set of user contexts $\mathcal{C} = \{c_1, \ldots, c_m\}$,

$\mathcal{C} \leftarrow \emptyset$, $\mathcal{E} := \infty$
$\mathcal{CI} \leftarrow CONTEXT(\mathcal{L})$ {extract context instances}
$\mathcal{C} \leftarrow AHC(\mathcal{CI}, \alpha)$ {apply Agglomerative Hierarchical Clustering on $\mathcal{CI}$. α a stopping threshold}
for all cluster $c_i \in \mathcal{C}$ **do**
  update its mean $m_i := NewMean(c_i)$
repeat
  $\mathcal{E}_{old} := \mathcal{E}$
  for all context instance $t_j \in \mathcal{CI}$ **do**
    assign t_j to its closest cluster $c^* \in \mathcal{C}$ such that $\forall c_i \in \mathcal{C}, MATCH(t_j, c_i) \leq MATCH(t_j, c^*)$
  for all cluster $c_i \in \mathcal{C}$ **do**
    update its mean $m_i \leftarrow NewMean(c_i)$
  compute $\mathcal{E} := \sum_{i=1}^{k} \sum_{t \in c_i} |MATCH(t, m_i)|^2$
until $\mathcal{E} \geq \mathcal{E}_{old}$
return $\mathcal{C}$

ements in P_u and contexts in C.

4.2.2 Contextual profile

In traditional RS, a user profile is a set of ratings $P_u = \{R_1, \ldots, R_n\}$, where each rating R_i is composed of a predicate pr_i and a rate (weight) r_i, i.e. $R_i = (pr_i, r_i)$. The rate r_i is a real number expressing the importance of the predicate pr_i for the user. pr_i is a triplet $< concept, operator, value >$, e.g. *Genre = 'Drama'*. *Concepts* may be *items* (content) that a user consumed, $features$ of these items or both of them.

Based on this profile, a contextualized user profile CP_u is defined as a set M of contextual mappings which relates user ratings to a set of contexts C:

$$CP_u = \{m_{ij}\,(R_i, c_j, s_{ij})\,|R_i \in P_u, c_j \in C, s_{ij} \in [-1, 1]\}$$

Each contextual mapping m_{ij} is defined by a rating R_i, a context c_j and a score s_{ij}. The score s_{ij} is a real number expressing the importance of taking into account the Rating R_i when the user interacts from the context c_j. Hereafter, we describe the way a contextual user profile is constructed using log files.

4.2.3 Service deployment on CARS

The user history H on which the contextualization service is based, is obtained by transforming the user log file L (figure 2). A sample of this history is presented in figure 3.

Id	Context- cluster	Content	Feature	C-Action	...
1	c_1	Item1	f1	+	...
2	c_1	Item2	f1	-	...
3	c_2	Item1	f2	-	...

Figure 3: a sample of user history H

User behaviour is captured in a relational table where each row is of the form: `<id, Context_id, Content_id, Feature, Action_type>` expressing that a user consumed a content (item) having some features (predicates) in a specific way `(Action_type)` within a given context `(context_id)`. `Action_type` specifies whether the action applied by a user when consuming the content is positive (e.g. Buy) or negative (e.g. Ignore). `Content_id` and `Features` are obtained by transforming the URI of the consumed content (which is a log file field) into more significant information (e.g. http://www.imdb.com/title/tt0448157/ becomes Title: *Hancock*, Genre:*Action, Comedy, Crime*). `Context_id` specifies the context cluster within which the content was consumed; it is computed by the context discovery service.

4.2.4 Contextualization Algorithm

The algorithm of contextual profile construction captures for each user rating (more precisely, predicate), its importance in each user context. Notice that, in each context, user activities may be of two kinds: positive (belongs to liked contents) and/or negative (relative to disliked contents), according to the type of actions a user applied on contents. Therefore the importance of each predicate must be captured in both positive and negative activities. As presented in [1], the importance of a predicate pr_i within a context c_j is captured by computing its frequency in this context. However, we claim that the frequency does not reflect the real importance of a predicate for one user. Thus, we propose to model the importance of each predicate pr_i for the positive (resp. negative) activity within a given context as an association rules of the form $\langle pr_i \rightarrow + \rangle$ (resp. $\langle pr_i \rightarrow - \rangle$). Then, the importance of pr_i is obtained by combining both the *support* (sp) and the *confidence* (cf) of its corresponding rules.

Many approaches can be used as merging function, we mention hereafter one possible way that consists in measuring the conviction of the rule.

$$conv(pr_i \rightarrow *) = \frac{1 - sp(*)}{1 - cf(pr_i \rightarrow *)} \quad (3)$$

Where $*$ takes one value among $\{+, -\}$ at a time.

Algorithm 2 shows the way contextual profiles are constructed based user's interaction histories.

Algorithm 2 Automatic contextual mappings discovery

Input: the initial user profile $P_u = \{p_1, \ldots, p_n\}$, the user behavior $\mathcal{H}$.
Output: the contextualized user profile CP_u

$CP_u \leftarrow \emptyset$, $cf := 0, sp := 0$
$\mathcal{C} \leftarrow CONTEXT(\mathcal{H})$
for all $R_i \in P_u$ **do**
  for all $c_j \in \mathcal{C}$ **do**
    compute $cf^+ := confidence(pr_i \rightarrow +, c_j, \mathcal{H}')$
    compute $sp^+ := support(pr_i \rightarrow +, c_j, \mathcal{H}')$
    compute $s_{ij}^+ := merge(cf^+, sp^+)$
    if $s_{ij}^+ \geq \gamma$ **then**
      $CP_u \leftarrow CP_u \cup (R_i, c_j, s_{ij}^+)$
    compute $cf^- := confidence(pr_i \rightarrow -, c_j, \mathcal{H}')$
    compute $sp^- := support(pr_i \rightarrow -, c_j, \mathcal{H}')$
    compute $s_{ij}^- := merge(cf^-, sp^-)$
    if $s_{ij}^- \geq \gamma$ **then**
      $CP_u \leftarrow CP_u \cup (R_i, c_j, -s_{ij}^-)$
return CP_u

The contextual profile constructed in algorithm 2 allows positioning each profile rating in each context through a score. The next section discusses the manner this score is exploited to make contextual recommendations.

5. ONLINE SERVICE: BINDING

This section details a runtime service that constitutes an answer to the last requirement imposed in section 2.

5.1 Service definition

The binding consists in identifying, at a runtime, user profile parts which are related to a given context.

$$OP_u : Binding(CP_u, c_i) \quad (4)$$

The profile binding takes as input the contextual user profile CP_u and the active context of the user c_i. It returns an operational profile OP_u which contains only profile elements which have to be considered by applications within c_i.

5.2 Service deployment on CARS

The Operational Profile is produced by filtering user ratings which aren't relevant to the active context, and by combining the remaining user ratings with contextual mapping scores.

Operational profile is always related to one context c_a (active context), it can be defined as being a set of contextual ratings (and not mappings): $OP_u = \{CR_{11}, \ldots, CR_{nk}\}$. Each contextual rating is of the form: $CR =< pr_i, w >$, and is derived from a contextual mapping $m_{ia}(R_i, c_a, s_{ia})$ with $R_i = pr_i \times r_i$. Notice that $w = aggregate(r_i, s_{ia})$ is the weight (contextual rating) of the predicate pr_i (e.g. $item_i$) within the context c_a.

The two reals r_i and s_{ia} have particular semantics. In fact, r_i represents the absolute importance of a given predicate pr_i. In other words, it is the preference that a user has on the predicate independently of contexts. However, a daily analysis of the user behaviour can reveal that the importance of the predicate pr_i on which r_i was expressed changes w.r.t contexts, leading to the definition of a contextual importance s_{ia}.

$$Aggregate(r_i, s_{ia}) = s_{ia} \times r_i \quad (5)$$

These two numeric can be aggregated with different manners, one possibility is given in equation 5 were r_i is multiplied by the contextual score s_{ia}. In this way, more the importance of the contextual score of a given pr_i is high, more its related rating r_i is preserved.

6. TOWARD A SERVICE-BASED CARS

Once contexts are learned and contextual profiles are constructed for each user in the RS, we show how runtime services (Binding and Matching) can be deployed in CARS to provide users with contextual recommendations. The next sections present the CARS algorithm, and a simple example scenario of its usage.

6.1 Contextual Top k neighbors

The key idea behind the CARS we propose is to base the Top k neighbors detection only on the profile parts relevant to a given context (i.e. operational profile) instead of on whole user profile. Top k neighbors of the active user (u_a) are the k most similar users to him in term of their profiles. This means that profiles of all users are filtered and adapted to the context of the active user before comparing them. Algorithm 3 details the process of computing the Top k contextual neighbors.

The algorithm is explained through the simple scenario given bellow.

Algorithm 3 Contextual Top K Neighbors

Input: $CP_{ua} = \{m_{11}, \ldots, m_{nm}\}$: Contextual active user profile, $\mathcal{U}$: set of all users, c_a: the active context, $\mathcal{C}$: the set of all contexts, $Item$: the item candidate to recommendation, k: the number of neighbors to consider, γ: the threshold.

Output: the TOP k neighbors

1: $CN \leftarrow \emptyset$ {Set of Candidate Neighbors}
2: $c^* := c \in \mathcal{C} | \forall c_i \in \mathcal{C}, MATCH(c_i, c_a) \leq MATCH(c^*, c_a)$
3: $OP_u := BIND(CP_u, c^*)$
4: **for all** $CP_{ui} \in \mathcal{U}$ **do**
5: **if** $m < (Item, *), *, * > \in CP_{ui}$ **then**
6: $OP_{ui} := BIND(CP_{ui}, c_a)$
7: $CN \leftarrow CN \cup \{OP_{ui}\}$
8: $TOPK \leftarrow \{OP_{ui} \in CN, i = \overline{1,k}\}$ such that $\forall OP_{uj} \in CN, OP_{uj} \notin TOPK, MATCH(OP_{uj}, OP_{ua}) \leq MATCH(OP_{ui}, OP_{ua})$
9: **return** $TOPK$

6.2 Aggregating Neighbor Ratings

Once Top k neighbors of the active user u_a are determined, ratings they gave to the item (It) to be recommended must be aggregated. The result of this aggregation allows deciding whether it is relevant or not to recommend It to the user u_a. There exist several techniques for aggregating these ratings [8, 6], the one we considered is the *Weighted Mean Aggregation* (equation 6) where the contribution of each neighbor rating is weighted with the similarity between this neighbor and the active user u_a.

$$Rate(u, It) = \alpha \sum_{u_i \in TopK} Match(u, u_i) \times Rate(u_i, It) \quad (6)$$

α is normalizing factor.

According to the value of the aggregated rating, the RS decides whether It could be or not recommended.

6.3 Simple scenario

The scenario below gives an intuition about the way CARS provides users with contextual recommendations. Figure 4 focuses on the recommender engine (RE) sketched in figure 1. Dashed arrows represent the inputs of the RE.

When recommendations are requested, explicitly by users (here John) or implicitly by the applications, RE computes them following three processes A, B, and C shown in fig 4.

First of all, a parser extracts the active context (c_a) of the user, and designates the candidate content to recommendation (step A1). In step B1, the active context (c_a) is matched with all context clusters in order to determine that to which c_a belongs (line 2 of the algorithm). This context cluster is used in step B2 for binding John's contextual profile and producing his operational profile (line 3). At the same time, RE looks for the candidate neighbors of John (process C: lines 4-7). A candidate neighbor is a user who rated the content to be recommended (filtering: step C1), in a context c_r similar to the active context c_a (step C2). In step C3, the contextual profile of each candidate user is bound to its corresponding context c_r, resulting in a set of operational profiles. Step A2 synchronizes the two processes B and C. It consists in determining the top k neighbors of John among the candidate users. This is done by capturing

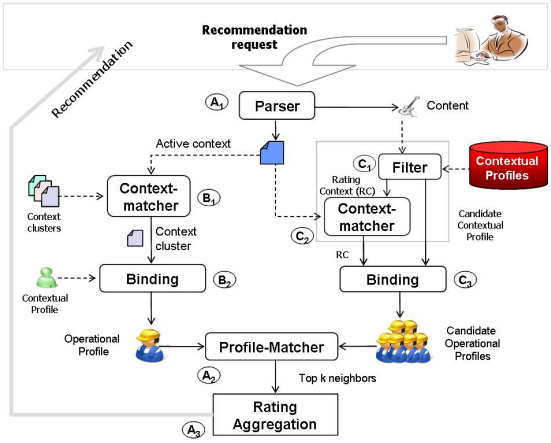


Figure 4: Contextual CF-RS

the similarity between each candidate profile and John profile; only top k similar profiles are retained (line 8). Step A3 aggregates ratings given to the recommended content by the top k returned neighbors. Finally, a decision is made to know whether it is interesting or not to recommend the candidate content (through a threshold for example).

7. RELATED WORK

Adomavicius et al. [4] discussed what a next generation of RS might be and proposed some possible extensions that will improve recommender capabilities such as the integration of contextual information into recommendation processes. In [3], a multidimensional approach is presented to incorporate the context into recommendations. The approach is based on ratings that are captured in OLAP cubes and which are sensitive to contextual information such as *Time, Place,* and *Accompanying people*. The same multidimensional cube-based approach was proposed to manage contextual preferences in the database field [13]. Contextual preferences are usually of the form: $Pref(item, c_1, \ldots, c_n) = w$, where c_i is a contextual attribute, and the weight $w \in [0, 1]$ expresses the interest a user has on the *item*. Notice that Contextual preferences were subject to extensive research in database field [12, 5, 2]. Hence, understanding these researches can help in designing more meaningful CARS. The difference between these approaches and our is that, in the latter one, contextual attributes are clustered to form a finite set of regular contexts of users (e.g. Lab context, Walking context) while, in the former, each instance of contextual attribute is considered itself as a context.

Inspired by human memory, Anand and Mobasher [6] proposed a contextual recommendation where users are modeled through a short-term memory (STM) which stores current interactions and a long-term memory (LTM) which stores previous interactions. Contextual Cues are generated from STM to extract relevant ratings from LTM according to the context (i.e. cues). These ratings are merged with the STM to provide user with contextual recommendations. The Binding service we presented has some similarities with this approach in the manner it computes the operational profile using both contextual profile (LTM) and active context (Cue). However, unlike Anand et al. who consider the interactional view of the context, we have considered the representational view in which the context is an information, stable, delineable, and separable from the activity [9]. More recently, there was attempts to define architectures for context-aware recommender [7]. However, author don't give details about the deployment of such architectures.

8. CONCLUSION

In this paper, we have proposed a set of personalization services that improve RS by introducing the notion of context. Among these services, two of them (Context Discovery and Contextualization) are design services and two others (Binding and Matching) are recommendation services. For each service, we have defined its operational semantics through one or several algorithms. Finally, we have shown how these services can be combined to form a CARS. Further research on this topic will concentrate on the evaluation of this approach by comparing traditional RS with CARS. To this end, a significant benchmark has to be defined.

9. REFERENCES

[1] S. Abbar, M. Bouzeghoub, D. Kostadinov, and S. Lopes. A contextualization service for a personalized access model. In *EGC*, pages 265–270, 2009.

[2] S. Abbar, M. Bouzeghoub, D. Kostadinov, S. Lopes, A. Aghasaryan, and S. Betge-Brezetz. A personalized access model: concepts and services for content delivery platforms. In *Proceedings of the 10th iiWas, Linz, Austria*, pages 41–47, 2008.

[3] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, 2005.

[4] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TKDE*, 17(6):734–749, 2005.

[5] R. Agrawal, R. Rantzau, and E. Terzi. Context-sensitive ranking. In *ACM SIGMOD*, pages 383–394, 2006.

[6] S. S. Anand and B. Mobasher. Contextual recommendation. *Discovering and Deploying User and Content Profiles. Berlin, Germany*, pages 142–160, 2007.

[7] L. Baltrunas. Exploiting contextual information in recommender systems. In *RecSys '08*, pages 295–298, New York, NY, USA, 2008. ACM.

[8] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Technical report, MSR-TR-98-12. Microsoft research, Redmond, WA 98052, 1998.

[9] P. Dourish. What we talk about when we talk about context. *Pers. Ubiq. Comput.*, pages 19–30, 2004.

[10] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 550 pages. ISBN 1-55860-489-8, 2000.

[11] B. B. Phillip M. Hallam-Baker. Extended log file format. Technical report, W3C Draf WD- logfile -960323. http://www.w3.org/TR/WD-logfile, 1996.

[12] K. Stefanidis and E. Pitoura. Fast contextual preference scoring of database tuples. In *EDBT, Nantes, France*, pages 344–355, 2008.

[13] K. Stefanidis, E. Pitoura, and P. Vassiliadis. A context-aware preference database system. *Interl. Journal of PCC*, pages 439–460, 2007.

"You May Also Like" Results in Relational Databases

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

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

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

ABSTRACT

In this position paper, we consider extending relational database systems with a recommendation functionality. In particular, we propose that, along with the results of each query, the user gets additional recommended results that we call *"You May Also Like"* or YMAL results. We discuss a suite of different approaches to computing YMAL results and exploit one that uses only the database content and the query results. Some preliminary evaluation results are also provided.

1. INTRODUCTION

The typical interaction of a user with a database system is by formulating queries. This interaction mode assumes that users are to some extent familiar with the content of the database and also have a clear understanding of their information needs. However, as databases get larger and accessible to a more diverse and less technically-oriented audience, a new "recommendation"-oriented form of interaction seems attractive and useful.

In this paper, motivated by the way recommenders work, we consider "recommending" to the users tuples not in the results of their queries but of potential interest. For instance, when asking for a *Woody Allen* movie, we could recommend a *Woody Allen* biography. When looking for drama movies produced in *England* with *Oscar* nominations, we could also recommend similar movies with *BAFTA* awards. Further, we may recommend what similar users have asked for in the past.

We call such results *"You May Also Like"* or YMAL results for short. YMAL results are useful because they let users see other tuples in the database that they may be unaware of.

We consider three fundamentally different approaches to computing YMAL results. The first one, termed *current-state*, uses the results of the current query and the database content. The second one, termed *history-based*, is similar to traditional recommendation systems. It uses the past history of user queries to suggest tuples that are results of either similar past queries or results of queries posed by similar users. The last one, termed *external sources*, considers using information from resources external to the database, such as the web.

We focus on the *current-state* approach and present a novel method for computing YMAL results. The method explores both the database schema by expanding the original query through joins with appropriate other relations and the database content through value correlations. We also report some preliminary evaluation results of this method.

Extending database queries with recommendations has also been suggested in two very recent works, namely [12] and [7]. FlexRecs [12] proposes a framework and a related engine for the declarative specification of the recommendation process. Here, we address a specific recommendation process, that of suggesting results relevant to a given user query and propose methods for generating them. Recommendations in [7] are more restricted than our YMAL results in that they are solely based on the past behavior of similar users. There is also some relation to query relaxation (e.g. [11]). Query relaxation addresses a different problem: extending the original query when there are not enough matching results.

The remainder of this paper is structured as follows. In Section 2, we provide a taxonomy of methods for producing YMAL results. In Section 3, we focus on a method that uses the current database instance and query results, while in Section 4, we present some preliminary results of this method. Section 5 summarizes related work and Section 6 concludes the paper.

2. YMAL RESULTS

Assume a database system $\mathcal{D}$ and a set of users $\mathcal{U}$ interacting with it by posing traditional select-project-join (SPJ) queries. Given a user $u \in \mathcal{U}$ and a query q, a typical database system returns a set of results $\mathcal{R}(q)$ in the form of tuples, possibly produced by joining several relations of $\mathcal{D}$. Besides $\mathcal{R}(q)$, we would like to locate and recommend to u a set of tuples that may also be of interest to the user. We call this set of tuples *"You May Also Like"* tuples or YMAL results for short. We denote this set as $\text{YMAL}(q, u)$. As a running example, we shall use the movies database shown in Figure 1.

In this paper, we propose a number of approaches to compute YMAL results. These approaches fall into three main categories:

1. *Current-state approaches*, that exploit the content and schema of the current query result and database instance.

PersDB 2009

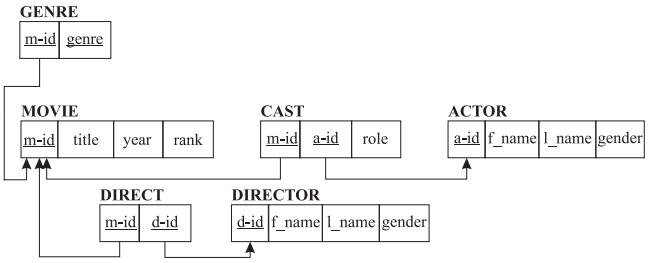


Figure 1: Movies database schema example.

2. *History-based approaches*, that exploit the history of previously submitted queries to the database system, e.g. by using query logs.

3. *External sources approaches*, that exploit resources external to the database, such as related published results and reports, relevant web pages, thesaurus or ontologies.

Figure 2 depicts a taxonomy of YMAL computation techniques. Next, we focus on each of these categories separately and present challenges and strategies towards the computation of YMAL results.

Current-state Approaches: We assume first, that there is no other information available other than a query q posed by a user u and its result $\mathcal{R}(q)$. Then, YMAL results can be computed based on either (i) *local analysis* of the intrinsic properties of the result $\mathcal{R}(q)$ or (ii) *global analysis* of the properties of the database $\mathcal{D}$. In both cases, we can exploit (i) the *content* and/or (ii) the *schema* of $\mathcal{R}(q)$ or $\mathcal{D}$ respectively.

There are many directions for computing YMAL results along these axes. For instance, in the local analysis approach, after $\mathcal{R}(q)$ has been computed, we examine the content of its tuples to locate common information patterns appearing in many of them. We then employ such information to retrieve and recommend tuples of the database that do not belong in $\mathcal{R}(q)$ but exhibit similar behavior. For example, assume that u poses a query to retrieve movie titles in which *Morgan Freeman* stars. Since *Morgan Freeman* often acts in detective roles, the relative attribute value *detective* appears many times in the result. Therefore, we recommend to u a number of movie titles in which other actors play the role of detective. Another option, in a schema-based approach, is to expand the tuples of the result through joins. Intuitively, in this way, we add extra, possibly useful information to the result and search for common patterns in the expanded result tuples. In our example, instead of presenting to u only the titles of *Morgan Freeman*'s movies, we may enhance the result with information about the corresponding genres. Based on the most frequent genres appearing in these expanded tuples, we recommend to u other movies of those genres.

In the global analysis case, we base YMAL computation on properties of $\mathcal{D}$. In the content approach, we rely on the correlation of specific attribute values as well as their selectivities. For example, when querying for *Walter Matthau* movies, we may also recommend a number of *Jack Lemmon* movies, since these two actors often star together. Correlation among relations can be used to direct the expansion of tuples in $\mathcal{R}(q)$ in a schema-based view of the problem.

Hybrid methods can also be applied by combining local and global analysis or content and schema information when processing a query result. Table 1 shows a taxonomy of the current-state approaches. We will examine further details of current-state approaches in Section 3.

History-based Approaches: History-based approaches assume that there is available information about the previous interactions of the users with the database, similar to traditional recommenders. In this respect, there are two alternatives: one could either log the results of the queries or the queries themselves. Technically, the two approaches are not equivalent, since the result of each query depends on the database instance, thus, the result of executing a logged query in the current database instance may differ significantly from the original result. Since logging results imposes significant overheads, for simplicity, in this position paper, we opt for logging queries.

Given a set of queries $\mathcal{Q}$ and a set of users $\mathcal{U}$, the utility function f: $\mathcal{Q} \times \mathcal{U} \rightarrow \mathcal{N}$, where $\mathcal{N}$ is a totally ordered set, measures the usefulness of a query $q \in \mathcal{Q}$ to a user $u \in \mathcal{U}$. We assume that the utility $f(q, u)$ is equal to the number of times user u has posed the query q.

Following the usual classification of recommendation systems, we distinguish between two different approaches: (i) *query-based* YMAL results (similar to *content-based* recommendations) and (ii) *user-based* YMAL results (similar to *collaborative* recommendations).

In the query-based approach, when a user u poses a query q, YMAL(q, u) includes results of the logged queries $q_i \in \mathcal{Q}$, that are the most similar to q, according to some similarity function $sim_q(q_i, q_j)$ between queries. For example, we may use:

$$sim_q(q_i, q_j) = \left|\mathcal{R}(q_i) \bigcap \mathcal{R}(q_j)\right|.$$

Using the utility function, we can represent each query q as a vector $(f(q, u_1), f(q, u_2), \ldots f(q, u_{|U|}))$. Then, for example, we can use as similarity, the inner product:

$$sim_q(q_i, q_j) = \sum_{k=1}^{|U|} f(q_i, u_k) f(q_j, u_k)$$

In the user-based approach, when a user u poses a query q, YMAL(q, u) includes results of queries posed by those users $u_j \in \mathcal{U}$ that exhibit the most similar behavior to u. Similar users are located via a similarity function $sim_u(u_i, u_j)$ between users, such as:

$$sim_u(u_i, u_j) = \left|\mathcal{Q}(u_i) \bigcap \mathcal{Q}(u_j)\right|$$

where $\mathcal{Q}(u_i)$ is the set of queries posed by u_i. Analogously to the queries, using the utility function, we can represent each user u as a vector $(f(q_1, u), f(q_2, u), \ldots f(q_{|Q|}, u))$. Then, for example, we can use as similarity, the inner product:

$$sim_u(u_i, u_j) = \sum_{k=1}^{|Q|} f(q_k, u_i) f(q_k, u_j)$$

In a hybrid approach, we present to u the results of the most similar queries to q out of those that were posed by similar users. Table 2 synopsizes history-based approaches.

Finally, we note that there is an important temporal dimension that needs to be considered. It is often the case that recent queries reflect better the current trends and interests

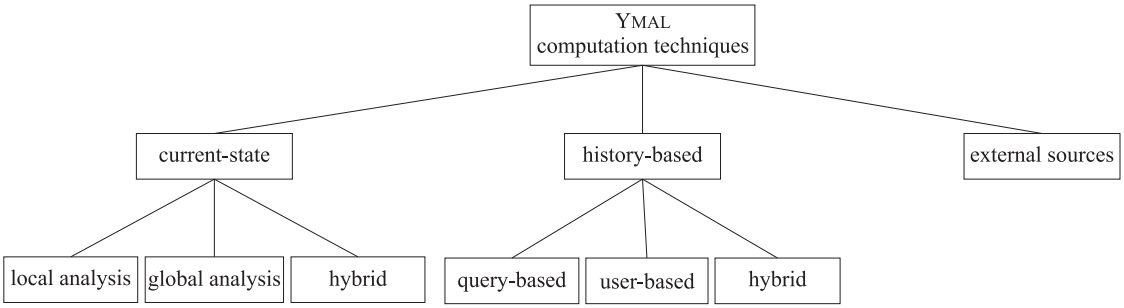


Figure 2: A taxonomy of YMAL computation techniques.

Table 1: A taxonomy of current-state approaches.

	Local analysis	**Global analysis**	**Hybrid analysis**
Content-based	Most frequent values in $\mathcal{R}(q)$	Most correlated values in $\mathcal{D}$	Combine frequent and correlated values
Schema-based	Direct joins through frequencies of values in expanded $\mathcal{R}(q)$	Direct joins through correlations among relations in $\mathcal{D}$	Direct joins through frequencies in expanded $\mathcal{R}(q)$ and correlations among relations in $\mathcal{D}$

of users, thus an aging scheme that gradually degrades the importance of queries in the log is into place.

External Sources Approaches: Up to now, we have discuss how we can locate and recommend YMAL results by exploiting intrinsic information of the database, such as correlation among attribute values and relations themselves. However, there are cases where relationships among data items are not captured in the database, even if present. Nowadays, a plethora of useful, well-organized information is available over the Web in the form of articles, reports and reviews in collectively maintained knowledge repositories, such as Wikipedia[1] and LibraryThing[2]. Information retrieved from such external sources can also be used for the computation of YMAL results.

For example, assume the database schema of Figure 1 and a query about *Sofia Coppola* movies. Using external information, we could recommend a number of *Francis Ford Coppola*'s movies, since he is the father of *Sofia Coppola*, a relationship that is not reflected in the schema. As another example, consider that a user poses a query that retrieves movies of various directors. Using an external source, interesting information may be inferred, e.g. that most of these directors are Asian. In this case, we could recommend other movies by Asian directors. Note that, the origin of directors cannot be found in the schema, so this correlation can be found only through external sources.

3. CURRENT-STATE APPROACHES

Current-state approaches explore the results $\mathcal{R}(q)$ of an SPJ query q to direct the computation of YMAL results for a user u. In this section, we will further examine such approaches and their application. First, we will focus on local analysis methods and then on global analysis ones.

[1]http://www.wikipedia.org
[2]http://www.librarything.com

Local Analysis: During local analysis of a query result $\mathcal{R}(q)$, we aim at discovering interesting patterns that we will later exploit to recommend YMAL results. Such patterns can be either found in the tuples of $\mathcal{R}(q)$ (content-based approach) or in the extended tuples produced by joining the tuples of the result with other tuples of the database (schema-based approach).

In this work, we view interesting patterns as frequently appearing attribute values, or combinations thereof. To quantify attribute values appearances in $\mathcal{R}(q)$, we define the *value-frequencies* matrix $M_{\mathcal{R}(q)}$. There is one row in $M_{\mathcal{R}(q)}$ for each attribute $A_1, \ldots, A_m$ of $\mathcal{R}(q)$ and one column for each distinct attribute value $v_1, \ldots, v_n$ appearing in its tuples. $M_{\mathcal{R}(q)}(i, j)$ contains the number of occurrences of v_j for A_i in $\mathcal{R}(q)$. As an example, consider a user that is interested in movie titles staring *Lee Phelps*. In Figure 3, we see a part of the related value-frequencies matrix. For ease of presentation, we depict only the five most frequent values for the attribute *Role*.

	Policeman	Detective	Cop	Bartender	Guard
Role	36	24	23	22	13

Figure 3: Value-frequencies matrix example.

Given a query q and the corresponding value-frequencies matrix $M_{\mathcal{R}(q)}$, our goal is to present to the user a set of YMAL results with cardinality p. Such results are computed with regards to the most frequent attribute values in $\mathcal{R}(q)$ as specified by $M_{\mathcal{R}(q)}$. In particular, we locate the k elements in $M_{\mathcal{R}(q)}$ with the highest values and, for each such element, we construct an appropriate SPJ query to retrieve interesting results. For clarity in notation, we also consider the matrix $M'_{\mathcal{R}(q)}$ for which $M'_{\mathcal{R}(q)}(i, j) = M_{\mathcal{R}(q)}(i, j)$ if $M_{\mathcal{R}(q)}(i, j)$ belongs to the k, $k > 0$, most frequent attribute values and $M'_{\mathcal{R}(q)}(i, j) = 0$ otherwise. Each element contributes a num-

Table 2: A taxonomy of history-based approaches.

Query-based approaches	**User-based approaches**	**Hybrid approaches**
Similarities among queries	Similarities among users	Similarities among both users and queries

ber of YMAL results based on the function F:

$$F(i,j) = \frac{M'_{\mathcal{R}(q)}(i,j)}{\sum_i \sum_j M'_{\mathcal{R}(q)}(i,j)} \cdot p$$

For the above example, let $p = 10$ and $k = 2$. Then, based on F, we will recommend six movies containing the role *Policeman* and four ones containing the role *Detective*.

When a schema-based approach is followed, we expand $\mathcal{R}(q)$ prior to constructing M, so that, additional interesting common patterns can be discovered. In our example, if we expand $\mathcal{R}(q)$ towards the *GENRE* relation, we discover other interesting information, such as, this actor mainly stars in *Drama* movies. For a specific query q and its result $\mathcal{R}(q)$, we expand $\mathcal{R}(q)$ towards all possible directions through the same number of join operations and construct the corresponding value-frequency matrices. Then, we select the matrix containing the most frequent value appearances and proceed with the YMAL computation as before, based on this matrix alone. We consider that patterns discovered after one join operation are more relevant than patterns discovered after two join operations and so on. For this reason, when selecting which matrix to use, we also take into account the corresponding number of needed join operations for each matrix and favor matrices with fewer joins.

Global Analysis: Global analysis aims at taking advantage of database properties during the recommendation of YMAL results. In this work, we consider that YMAL computation is guided by certain statistics maintained for our database $\mathcal{D}$. Such statistics include the correlation among the various attribute values of the database and the correlation among the various relations in $\mathcal{D}$.

We define the *value-correlation* matrix V that captures the correlation between pairs of distinct attribute values in each database relation. V is a $(x \times y \times y)$ matrix, where x is the number of attributes in $\mathcal{D}$ and y is the number of distinct attribute values. Given a query q, we consult the matrix V to locate attribute values correlated to the selection predicates of q. We select the k attribute values that are the most correlated to q. The more correlated such a value is, the more YMAL results it will contribute. This can be calibrated via the use of a function similar to F that is defined based on V instead of M.

The correlation among the relations of $\mathcal{D}$ can be used to direct the joining operations in a schema-based approach. Such correlations are captured in the $(z \times z)$ *relation-correlation* matrix A, where z is the number of relations in $\mathcal{D}$. For example, assuming that the relation *CAST* is strongly correlated with the relation *ACTOR*, then, when querying for specific actor names we could present roles that these actors have portrayed.

Hybrid Analysis: Each of the methods described above exploits different properties: local analysis is based on the actual results of a query, while global analysis depends on the whole database. As a next step, we can combine the advantages of both approaches in a hybrid method that exploits both local and global properties. A hybrid content-based approach is to use attribute values that are both frequent and strongly correlated. To calibrate the importance of each factor, we rely on a weighted function. Similarly, in a schema-based approach, the joining of $\mathcal{R}(q)$ with other database relations is directed using both the correlations among the relations of the database, as well as, frequent appearances of attribute values in those relations.

Table 3: Relation cardinalities.

Relation name	**Cardinality**
GENRE	109.261
MOVIE	70.266
CAST	1.266.462
ACTOR	322.467
DIRECT	109.226
DIRECTOR	152.533

4. EVALUATION

Our evaluation objective is to demonstrate the effectiveness of our approach for current-state methods. In particular, to show the usefulness of YMAL results, we will present for representative queries both their $\mathcal{R}(q)$ and YMAL(q, u) results. For our experiments, we use a real movie dataset [1]. The schema of the database is depicted in Figure 1, while Table 3 shows the cardinalities of the relations.

Local analysis: To experiment with local analysis YMAL computation, we use the following query:

```
q1 :  select  *
      from    MOVIE, CAST, ACTOR
      where   MOVIE.m-id = CAST.m-id and
              CAST.a-id = ACTOR.a-id and
              ACTOR.f_name = 'Lee' and
              ACTOR.l_name = 'Phelps';
```

A subset of $\mathcal{R}(q_1)$ is shown in Figure 4. The part (*372839 - Lee - Phelps - M*) is common in all tuples of $\mathcal{R}(q_1)$ due to the query selection conditions, therefore, we exclude its attribute values from the construction of $M_{\mathcal{R}(q_1)}$. For $k = 2$, the two most common values in the 394 tuples of $\mathcal{R}(q_1)$ are the values *Policeman* and *Detective* of the attribute *Role* (Figure 3). For $p = 3$, *Policeman* and *Detective* contribute two and one YMAL results respectively, when the content-based approach is followed (Figure 4).

$\mathcal{R}(q_1)$ **results**										
m-id	**title**	**year**	**rank**	**a-id**	**m-id**	**role**	**a-id**	**f_name**	**l_name**	**gender**
4619	Abbott and Costello in Hollywood	1945	5.6	372839	4619	Detective	372839	Lee	Phelps	M
218015	Money to Loan	1939	6.3	372839	218015	Policeman	372839	Lee	Phelps	M
46730	Bride Came C.O.D., The	1941	6.8	372839	46730	Policeman	372839	Lee	Phelps	M
330384	Thin Man Goes Home, The	1945	7	372839	330384	Policeman	372839	Lee	Phelps	M
31821	Beast From 20,000 Fathoms, The	1953	6.3	372839	31821	Cop	372839	Lee	Phelps	M
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ymal(q_1, u) **results: Content-based Approach**										
m-id	**title**	**year**	**rank**	**a-id**	**m-id**	**role**	**a-id**	**f_name**	**l_name**	**gender**
323813	Talented Mr. Ripley, The	1999	7	411152	323813	**Policeman**	411152	Manuel	Ruffini	M
195807	Love Letter	1998	8.2	420874	195807	**Policeman**	420874	Bsaku	Satoh	M
155070	I, Robot	2004	6.9	297823	155070	**Detective**	297823	Craig	March	M

Figure 4: Content-based local analysis for q_1.

When the schema-based approach is employed, the possible directions for expansion are the relations *GENRE* and *DIRECT*. The most common patterns are observed when $\mathcal{R}(q_1)$ is expanded towards the *GENRE* relation and are the values *Drama* and *Comedy* that appear 216 and 120 times respectively. The expanded tuples of $\mathcal{R}(q_1)$ and the recommended YMAL results in this case, for $k = 2$ and $p = 3$, are the ones shown in Figure 5.

Global analysis: To experiment with content-based, global analysis YMAL computation, we use the following query that retrieves romance movies:

```
q2 :  select  *
      from    MOVIE, GENRE
      where   MOVIE.m-id = GENRE.m-id and
              GENRE.genre = 'Romance';
```

In our dataset, the attribute value *Romance* appears along with other genres for the same movies as many times as shown below:

Drama (2801)	Comedy (2398)	Musical (538)
Action (351)	Adventure (323)	Thriller (267)
Fantasy (263)	Crime (263)	Family (234)
War (199)	Short (162)	Mystery (131)

We use again $k = 2$ and $p = 3$. As we can see above, the two mostly correlated values to *Romance* are *Drama* and *Comedy*. Therefore, two drama movies and a comedy one will be recommended. We omit the relative figure due to space limitations.

Consider now the query:

```
q3 :  select  *
      from    MOVIE, DIRECT, DIRECTOR
      where   MOVIE.m-id = DIRECT.m-id and
              DIRECT.d-id = DIRECTOR.d-id and
              DIRECTOR.f_name = 'Steven' and
              DIRECTOR.l_name = 'Spielberg';
```

In our database, the relation that is most correlated to *MOVIE* is *GENRE*. Therefore, when computing YMAL results using the schema-based approach, we enhance $\mathcal{R}(q_3)$ with information about the genres of Steven Spielberg's movies.

5. RELATED WORK

In this paper, we have proposed extending relational database systems with recommendation functionality in the form of YMAL results. In general, recommendation methods are categorized into: (i) *content-based*, that recommend items similar to those the user has preferred in the past (e.g. [16, 14]), (ii) *collaborative*, that recommend items that similar users have liked in the past (e.g. [10, 6]) and (iii) *hybrid*, that combine content-based and collaborative ones (e.g. [4, 5]). [3] provides a comprehensive survey of the current generation of recommendation systems. Several extensions have also been proposed, such as employing multi-criteria ratings [2] and extending the typical recommendation systems beyond the two dimensions of users and items to include further contextual information [15].

In terms of relating recommendations and databases, there are two very recent works [12, 7]. [12] provides a general framework and an engine for the declarative specification of the recommendation process over structured data. In this paper, we focus on the specific recommendation process of computing YMAL results related to a specific user query. The recommendation process in [12] is specified through a series of interconnected operators, which apart from the traditional relational operators, includes also a number of specific to the recommendation process operators, such as the *recommend* operator, that recommends a set of tuples of a specific relation with regards to their relationship with the tuples of another relation. In our approach, we rely on typical relational algebra operators. In [7], the focus is on recommending SQL queries to the users of a database. The proposed method is based on "session summaries", i.e. the set of tuples that contributed to some result for queries imposed by the user in the current session. Given a session summary for a user u, the purpose is to compute a prediction summary of tuples that may be of interest to the user and then locate a number of queries able to retrieve the tuples in it. The prediction summary is computed based on the session summary of u and other users similar to u. The suggested queries are retrieved from a pool of past queries submitted by the users. Recommendations in [7] are more restricted than our YMAL results, since they are solely based on the past behavior of similar users.

There is also some relation with *query reformulation*, where a query is relaxed or restricted when the number of results of the original query are too few or too many respectively, and

Expanded $\mathcal{R}(q_1)$ results												
m-id	**title**	**year**	**rank**	**a-id**	**m-id**	**role**	**a-id**	**f_name**	**l_name**	**gender**	**m-id**	**genre**
4619	Abbott and Costello in Hollywood	1945	5.6	372839	4619	Detective	372839	Lee	Phelps	M	4619	Comedy
218015	Money to Loan	1939	6.3	372839	218015	Policeman	372839	Lee	Phelps	M	218015	Drama
46730	Bride Came C.O.D., The	1941	6.8	372839	46730	Policeman	372839	Lee	Phelps	M	46730	Comedy
330384	Thin Man Goes Home, The	1945	7	372839	330384	Policeman	372839	Lee	Phelps	M	330384	Drama
31821	Beast From 20,000 Fathoms, The	1953	6.3	372839	31821	Cop	372839	Lee	Phelps	M	31821	Sci-Fi
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Ymal(q_1, u) results: Schema-based Approach												
256348	Pinocchio	2002	4	36868	256348	Pinocchio	36868	Roberto	Benigni	M	256348	**Comedy**
34306	Berlin Berlin	1998	9.7	426546	34306	Sammy	426546	Asad	Schwarz	M	34306	**Drama**
218345	Monster	200	7.4	91661	218345	Newscaster	91661	Jim R.	Coleman	M	218345	**Drama**

Figure 5: Schema-based local analysis for q_1.

with *automatic result ranking*, where the results of a query are restricted to the top-ranked ones, when the number of results of the original query is very large. Such approaches usually use the frequency of values of specific attributes in the database to restrict or expand the result set, which is also the basic idea of our *current-state* approaches in computing YMAL results.

A framework for relaxing queries involving numerical conditions in selection and join predicates is proposed in [11], while the relaxation algorithm proposed in [9] produces a relaxed query for a given query range and a desired cardinality of the result set. To estimate the result size, the algorithm uses multi-dimensional histograms. [13] studies the problem of query refinement through transformations of the selection query predicates. Transformations aim at either relaxing the query predicates in order to increase the result cardinality or contracting the query predicates in order to decrease the result cardinality. A systematic approach for the automatic ranking of query results is proposed in [8]. To estimate the rank of a result tuple, they use both workload and data analysis; this is similar to the *history-based* and *current-state* approaches respectively. The main difference is that we consider recommending tuples not in the result set, whereas [8] consider ranking the tuples in the result set. The operator *frequent co-occurring term* [17] can also be used to direct query refinement in relational keyword search. Given a keyword query q, this operator returns a set of terms that appear frequently in the result of q and none of them is contained in q. These are the terms that can be employed by users to refine their queries.

6. CONCLUSIONS

In this paper, we presented a first approach to computing YMAL results and organized the various alternatives into categories. There is a number of open issues for research. In terms of current-state YMAL computation, we could exploit information about the importance of each relation attribute. For history-based YMAL computation, we could explore techniques based on the logging of query results or statistics about query results instead of logging queries. Also, in this work, we looked into selecting YMAL results based on similarity. We could also consider other criteria, such as presenting to the users novel, fresh or diverse information [18].

Acknowledgment

We would like to thank Yufei Tao for providing the dataset used in this work.

7. REFERENCES

[1] *Internet Movies Database*. Available at www.imdb.com.
[2] G. Adomavicius and Y. Kwon. New recommendation techniques for multicriteria rating systems. *IEEE Intelligent Systems*, 22(3):48–55, 2007.
[3] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.
[4] M. Balabanovic and Y. Shoham. Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.
[5] C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720, 1998.
[6] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.
[7] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis. Query recommendations for interactive database exploration. In *SSDBM*, 2009.
[8] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic information retrieval approach for ranking of database query results. *ACM Trans. Database Syst.*, 31(3):1134–1168, 2006.
[9] A. Kadlag, A. V. Wanjari, J. Freire, and J. R. Haritsa. Supporting exploratory queries in databases. In *DASFAA*, pages 594–605, 2004.
[10] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997.
[11] N. Koudas, C. Li, A. K. H. Tung, and R. Vernica. Relaxing join and selection queries. In *VLDB*, pages 199–210, 2006.
[12] G. Koutrika, B. Bercovitz, and H. Garcia-Molina. Flexrecs: Expressing and combining flexible recommendations. In *SIGMOD*, 2009.
[13] C. Mishra and N. Koudas. Interactive query refinement. In *EDBT*, pages 862–873, 2009.
[14] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. *CoRR*, cs.DL/9902011, 1999.
[15] C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE Trans. Knowl. Data Eng.*, 20(11), 2008.
[16] M. J. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.
[17] Y. Tao and J. X. Yu. Finding frequent co-occurring terms in relational keyword search. In *EDBT*, pages 839–850, 2009.
[18] C. Yu, L. V. S. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *EDBT*, pages 368–378, 2009.