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.

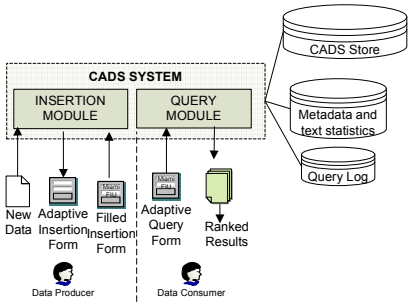

Figure 2: CADS Workflow.

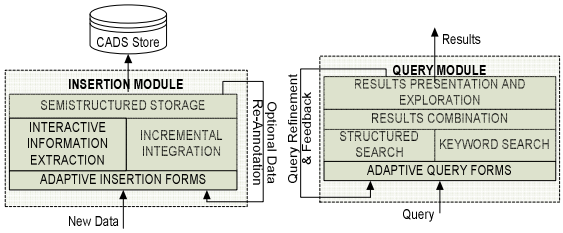

Figure 3: Architecture of Insertion and Query Modules.

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.

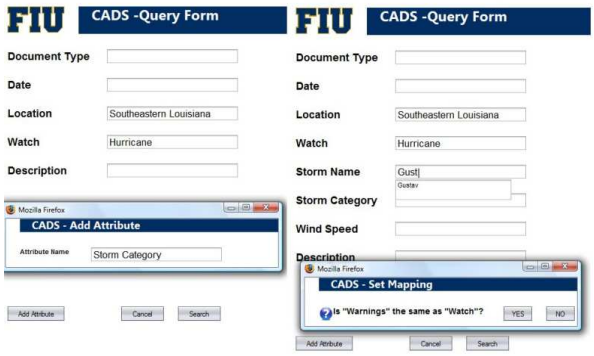

Figure 5: Adaptive Query Form.

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.

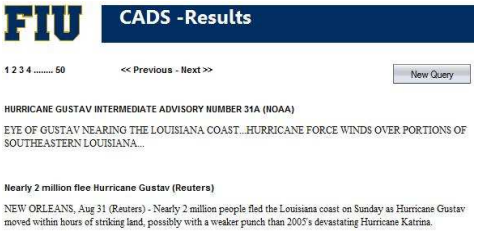

Figure 6: Query Results.

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.

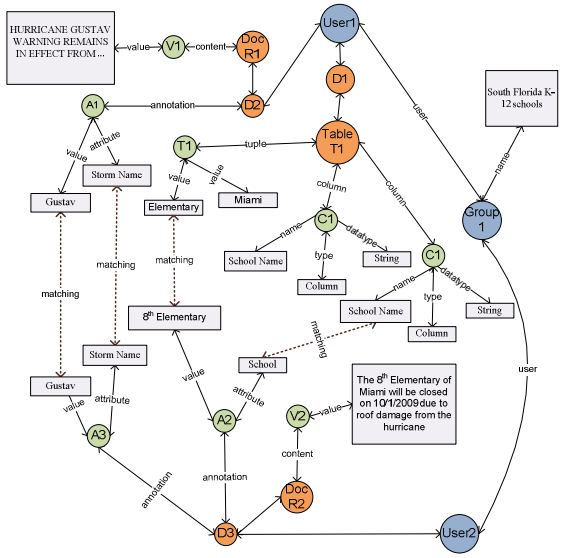

Figure 7: CADS graph

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