Authority-Based Keyword Search in Databases

Vagelis Hristidis
Florida International University
Miami, FL 33199
vagelis@cis.fiu.edu
and
Heasoo Hwang
Computer Science & Engineering
UC, San Diego
La Jolla, CA 92093
heasoo@cs.ucsd.edu
and
Yannis Papakonstantinou
Computer Science & Engineering
UC, San Diego
La Jolla, CA 92093
yannis@cs.ucsd.edu

Our system applies authority-based ranking to keyword search in databases modeled as labeled graphs. Three ranking factors are used: the relevance to the query, the specificity and the importance of the result. All factors are handled using authority-flow techniques that exploit the link-structure of the data graph, in contrast to traditional Information Retrieval. We address the performance challenges in computing the authority flows in databases by using precomputation and exploiting the database schema if present. We conducted user surveys and performance experiments on multiple real and synthetic datasets, to assess the semantic meaningfulness and performance of our system.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.2.8 [**Database Management**]: Database Applications

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: authority flow, ranking, PageRank, specificity, quality experiments

Vagelis Hristidis was partly supported by NSF grant IIS-0534530.

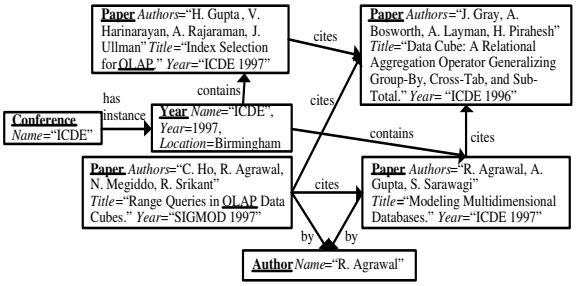

Fig. 1. A subset of the DBLP graph

1. INTRODUCTION

PageRank [Brin and Page 1998] is an excellent tool to rank the global importance of the pages of the Web. However, PageRank measures the global importance of the pages, independently of a keyword query. More recent works [Haveliwala 2002; Richardson and Domingos 2002] apply PageRank to estimate the relevance of pages to a keyword query. We appropriately extend and modify PageRank to perform keyword search in databases for which there is a natural flow of authority between their objects (e.g., bibliographic, biological [Raschid et al. 2006; Shafer et al. 2006] or complaints databases as we explain below).

Given a keyword query, we rank the results according to three factors: (a) the relevance to the query, (b) the specificity, and (c) the global importance of the result. All factors are handled using authority-flow techniques that exploit the link-structure of the data graph, in contrast to traditional Information Retrieval. The relevance is computed using the ObjectRank metric [Balmin et al. 2004] which is a keyword-specific adaptation of PageRank to databases. The specificity is computed using Inverse ObjectRank metric, which is, to the best of our knowledge, the first link-based specificity metric. Finally, the global importance is computed using Global ObjectRank, which is the keyword-independent version of ObjectRank. We show how these factors are combined to reach the final results ranking.

ObjectRank Consider the example of Figure 1, which illustrates a small subset of the DBLP database in the form of a labeled graph (author, conference and year nodes except for "R. Agrawal", "ICDE" and "ICDE 1997" respectively are omitted to simplify the figure). Schema graphs, such as the one of Figure 4, describe the structure of database graphs. Given a keyword query, e.g. the single-keyword query "OLAP", ObjectRank sorts the database objects by their relevance with respect to the user-provided keywords. Figure 2 illustrates the top-10 "OLAP" papers produced by our online demo available on the Web at two mirror sites, `http://www.db.ucsd.edu/ObjectRank` and `http://dbir.cis.fiu.edu/BibObjectRank`. Notice that many entries (the "Data Cube" and the "Modeling Multidimensional Databases" papers in Figure 1) of the top-10 list do not contain the keyword "OLAP" ("OLAP" is not even contained in their abstracts) but they clearly constitute important papers in the OLAP area, since they may be referenced by other papers of the OLAP area or may have been written by authors who have written other important "OLAP" papers.

Conceptually, the ranking is produced in the following way: Myriads of random surfers are initially found at the objects containing the keyword "OLAP", which we call the base

1 **Implementing Data Cubes Efficiently.** *SIGMOD Conference* 1996. Venky Harinarayan, Anand Rajaraman, Jeffrey D. Ullman
2 **An Overview of Data Warehousing and OLAP Technology.** *SIGMOD Record* 1997. Surajit Chaudhuri, Umeshwar Dayal
3 **Index Selection for OLAP.** *ICDE* 1997. Himanshu Gupta, Venky Harinarayan, Anand Rajaraman, Jeffrey D. Ullman
On the Computation of Multidimensional Aggregates. *VLDB* 1996. Sameet Agarwal, Rakesh Agrawal, Prasad Deshpande, Ashish
4 Gupta, Jeffrey F. Naughton, Raghu Ramakrishnan, Sunita Sarawagi
Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. *ICDE* 1996. Adam
5 Bosworth, Jim Gray, Andrew Layman, Hamid Pirahesh
6 **Summarizability in OLAP and Statistical Data Bases.** *SSDBM* 1997. Hans-Joachim Lenz, Arie Shoshani
7 **Modeling Multidimensional Databases.** *ICDE* 1997. Rakesh Agrawal, Ashish Gupta, Sunita Sarawagi
8 **OLAP, Relational, and Multidimensional Database Systems.** *SIGMOD Record* 1996. George Colliat
9 **OLAP and Statistical Databases: Similarities and Differences.** *PODS* 1997. Arie Shoshani
10 **OLAP and Statistical Databases: Similarities and Differences.** *CIKM* 1996. Arie Shoshani

Fig. 2. Top 10 papers on "OLAP" returned by ObjectRank

set, and then they traverse the database graph. In particular, at any time step a random surfer is found at a node and either (i) makes a move to an adjacent node by traversing an edge, or (ii) jumps randomly to an "OLAP" node without following any of the links. The probability that a particular traversal happens depends on multiple factors, including the type of the edge (in contrast to the Web link-based search systems [Brin and Page 1998; Haveliwala 2002; Richardson and Domingos 2002]). These factors are depicted in an authority transfer schema graph. Figure 5 illustrates the authority transfer schema graph that corresponds to the setting that produced the results of Figure 2. Assuming that the probability that the surfer moves back to an "OLAP" node is 15% (damping factor–random jump probability–[Brin and Page 1998]), the collective probability to move to a referenced paper is up to $85\% \times 70\%$ (70% is the authority transfer rate of the citation edge as we explain below), the collective probability to move to an author of the paper is up to $85\% \times 20\%$, the probability to move from the paper to the forum where the paper appeared is up to $85\% \times 10\%$, and so on. As is the case with the PageRank algorithm as well, as time goes on, the expected percentage of surfers at each node v converges (Section 2) to a limit $r(v)$. Intuitively, this limit is the ObjectRank of the node.

An alternative way to conceive the intuition behind ObjectRank is to consider that authority/importance flows in the database graph in the same fashion that [Kleinberg 1999] defined authority-based search in arbitrary graphs. Initially the "OLAP" authority is found at the objects that contain the keyword "OLAP". Then authority/importance flows, following the rules in the authority transfer schema graph, until an equilibrium is established that specifies that a paper is authoritative if it is referenced by authoritative papers, is written by authority authors and appears in authority conferences. Vice versa, authors and conferences obtain their authority from their papers. Notice that the amount of authority flow from, say, paper to cited paper or from paper to author or from author to paper, is arbitrarily set by a domain expert and reflects the semantics of the domain. For example, common sense says that in the bibliography domain a paper obtains very little authority (or even none) by referring to authoritative papers. On the contrary it obtains a lot of authority by being referred by authoritative papers.

Global ObjectRank is query-independent and is obtained by placing all nodes of the data graph in the base set.

Inverse ObjectRank Ranking solely by ObjectRank can be problematic, since general-content nodes may be ranked higher than nodes with content specific to the query. For example, consider the publications database of Figure 3, where edges denote citations (edges start from citing and end at cited paper), and the keyword query "Sorting". Then, using ObjectRank the "Access Path Selection in a Relational Database Management System" paper would be ranked highest, because it is cited by four papers containing "sorting" (or "sort"). The "Fundamental Techniques for Order Optimization" paper would be ranked

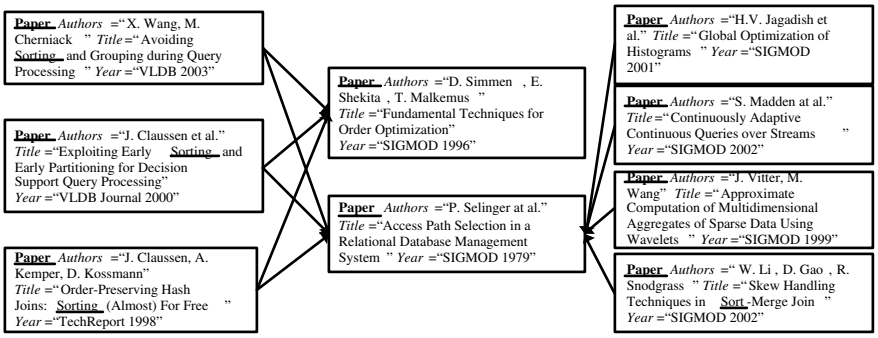

Fig. 3. Instance of a Publications Database

second, since it is cited by only three "sorting" papers. This is unintuitive since the "Access Path Selection" paper has general content while the "Fundamental Techniques for Order Optimization" paper is more focused (specific). The latter paper should be ranked higher because it is mostly cited by "sorting" papers, whereas the former paper is also cited by many (the three papers on the top right) papers irrelevant to "sorting". This lack of specificity can also be viewed as a topic-drift problem.

Google uses (to the best of our knowledge) IR techniques based on the content of the Web pages (e.g., document length), which ignore the link-structure of the labeled graph (i.e., the Web). Clearly, IR specificity metrics (e.g., document length) are not adequate since a longer document may be more specific than a shorter one for a particular query. However, IR metrics can be used in conjunction to Inverse ObjectRank to measure specificity.

Inverse ObjectRank is a keyword-specific metric of specificity, based on the link-structure of the data graph. In particular, given a keyword w, the Inverse ObjectRank score $p^w(v)$ of node v shows how specific v is with respect to w. In terms of the random surfer model, $p^w(v)$ is the probability that starting from v and following the edges on the opposite direction we are on a node containing w at a specific point in time. As is the case for ObjectRank, the random surfer at any time step may get bored and go back to v.

Keyword search in databases has some unique characteristics, which make the straightforward application of the random walk model as described in previous work [Brin and Page 1998; Haveliwala 2002; Richardson and Domingos 2002] inadequate. First, every database has different semantics, which we can use to improve the quality of the keyword search. In particular, unlike the Web, where all edges are hyperlinks, the database schema exhibits the types of edges, and the attributes of the nodes. Note that previous works [Richardson and Domingos 2002; Chakrabarti et al. 1998] assign weights on the edges of the data graph according to the relevance of the incident nodes' text to the keywords. In contrast, we assign authority transfer rates on the schema graph, which captures the semantics of the database, since the relevance factor is reflected in the selection of the base set. Using the schema we specify the ways in which authority flows across the nodes of the database graph. For example, the results of Figure 2 were obtained by annotating the schema graph of Figure 4 with the authority flow information that appears in Figure 5.

Furthermore, previous work [Brin and Page 1998; Haveliwala 2002; Richardson and Domingos 2002] assumes that, when calculating the global importance (in our framework we make a clear distinction between the global importance of a node and its relevance to

a keyword query), the random surfer has the same probability to start from any page p of the base set (we call this probability *base ObjectRank* of p). However, this is not true for every database. For example, consider a product complaints database (Figure 14). In this case, we represent the business value of a customer by assigning to his/her node a base ObjectRank proportional to his/her total sales amount.

Another novel property of ObjectRank is adjustability, which allows for the tuning of the system according to the domain- and/or user-specific requirements. For example, for a bibliographic database, a new graduate student desires a search system that returns the best reading list around the specified keywords, whereas a senior researcher looks for papers closely related to the keywords, even if they are not of a high quality. These preference scenarios are made possible by adjusting the weight of the global importance versus the relevance to the keyword query. Changing the damping factor d offers another calibration opportunity. In particular, larger values of d favor nodes pointed by high-authority nodes, while smaller values of d favor nodes containing the actual keywords (that is, nodes in the base set). The handling of queries with multiple keywords offers more flexibility to the system as we describe in Section 4. For example, we may want to assign a higher weight to the relevance of a node to an infrequent keyword.

On the performance level, calculating the ObjectRank, Inverse ObjectRank and Global ObjectRank values in runtime is a computationally intensive operation, especially given the fact that multiple users query the system. This is resolved by precomputing inverted indexes where for each keyword we have a sorted lists of the nodes with non-trivial scores for this keyword. During run-time we employ the *Threshold Algorithm* [Fagin et al. 2001] to efficiently combine the lists. However, our approach induces the cost of precomputing and storing the inverted index. Regarding the space requirements, notice that the number of keywords of a database is typically less than the number of users in a personalized search system [Jeh and Widom 2003]. Furthermore, we do not store nodes with ObjectRank below a threshold value (chosen by the system administrator), which offers a space versus precision tradeoff. In Section 8 we show that the index size is small relative to the database size for two bibliographic databases.

Regarding the index computation, we present and experimentally evaluate two classes of optimizations. First, we exploit the structural properties of the database graph. For example, if we know that the objects of a subgraph of the schema form a Directed Acyclic Graph (DAG), then given a topological sort of the DAG, there is an efficient straightforward one-pass ObjectRank evaluation. We extend the DAG case by providing an algorithm that exploits the efficient evaluation for DAGs in the case where a graph is "almost" a DAG in the sense that it contains a large DAG subgraph. In particular, given a graph G with n nodes, which is reduced to a DAG by removing a small subset of m nodes, we present an algorithm which reduces the authority calculation into a system of m equations - as opposed to the usual system of n equations. Furthermore, we present optimization techniques when the data graph has a small vertex cover, or if it can be split into a set of subgraphs and the connections between these subgraphs form a DAG.

Second, notice that the naive approach would be to calculate each keyword-specific ObjectRank (the same applies for Inverse ObjectRank) separately. We have found that it is substantially more efficient to first calculate the Global ObjectRank, and use these scores as initial values for the keyword-specific computations. This accelerates convergence, since in general, objects with high Global ObjectRank, also have high keyword-specific Objec-

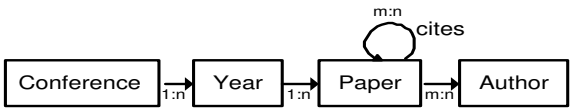

Fig. 4. The DBLP schema graph.

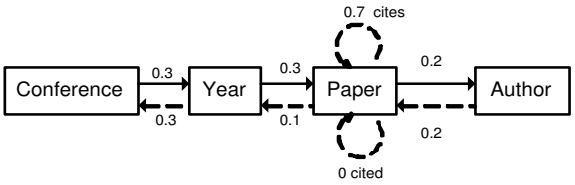

Fig. 5. The DBLP authority transfer schema graph.

tRanks. Furthermore, we show how storing a prefix of the inverted lists allows the faster calculation of the ObjectRanks of all nodes.

The semantic and performance contributions of this paper are evaluated using two user surveys and a detailed experimental evaluation respectively. We have implemented a web interface, available on the Web, to query the DBLP database using the ObjectRank technique. A set of calibrating parameters are provided to the user.

The essential formal background on PageRank and authority search is presented in Section 2. Section 3 presents the problem and the framework. Section 4 presents the semantics of ObjectRank and Inverse ObjectRank, as well as ways to combine them. Section 5 describes the system's architecture and the online demo. The algorithms used to calculate ObjectRank are presented in Section 6 and are experimentally evaluated in Section 8. We present the results of two user surveys in Section 7. Related work is discussed in Section 9. Finally, we conclude in Section 10.

2. BACKGROUND

We describe next the essentials of PageRank and authority-based search, and the random surfer intuition. Let (V, E) be a graph, with a set of nodes $V = \{v_1, \ldots, v_n\}$ and a set of edges E. A surfer starts from a random node (web page) v_i of V and at each step, he/she follows a hyperlink with probability d or gets bored and jumps to a random node with probability $1 - d$. The PageRank value of v_i is the probability $r(v_i)$ that at a given point in time, the surfer is at v_i. If we denote by $\mathbf{r}$ the vector $[r(v_1), \ldots, r_(v_i), \ldots, r(v_n)]^T$ then we have

$$\mathbf{r} = d\mathbf{A}\mathbf{r} + \frac{(1-d)}{|V|}\mathbf{e} \tag{1}$$

where $\mathbf{A}$ is a $n \times n$ matrix with $A_{ij} = \frac{1}{OutDeg(v_j)}$ if there is an edge $v_j \rightarrow v_i$ in E and 0 otherwise, where $OutDeg(v_j)$ is the outgoing degree of node v_j. Also, $\mathbf{e} = [1, \ldots, 1]^T$.

The above PageRank equation is typically precomputed before the queries arrive and provides a global, keyword-independent ranking of the pages. Instead of using the whole set of nodes V as the *base set*, i.e., the set of nodes where the surfer jumps when bored, one can use an arbitrary subset S of nodes, hence increasing the authority associated with the nodes of S and the ones most closely associated with them. In particular, we define a *base*

vector $\mathbf{s} = [s_0, \ldots, s_i, \ldots, s_n]^T$ where s_i is 1 if $v_i \in S$ and 0 otherwise. The PageRank equation is then

$$\mathbf{r} = d\mathbf{A}\mathbf{r} + \frac{(1-d)}{|S|}\mathbf{s} \tag{2}$$

Regardless of whether one uses Equation 1 or Equation 2 the PageRank algorithm solves this fixpoint using a simple iterative method, where the values of the (k+1)-th execution are calculated as follows:

$$\mathbf{r}^{(k+1)} = d\mathbf{A}\mathbf{r}^{(k)} + \frac{(1-d)}{|S|}\mathbf{s} \tag{3}$$

The algorithm terminates when $\mathbf{r}$ converges, which is guaranteed to happen under very common conditions [Motwani and Raghavan 1995]. In particular, the authority flow graph needs to be irreducible (i.e., (V, E) be strongly connected) and aperiodic. The former is true due to the damping factor d, while the latter happens in practice.

The notion of the base set S was suggested in [Brin and Page 1998] as a way to do personalized rankings, by setting S to be the set of bookmarks of a user. In [Haveliwala 2002] it was used to perform topic-specific PageRank on the Web. We take it one step further and use the base set to estimate the relevance of a node to a keyword query. In particular, the base set consists of the nodes that contain the keyword as explained next.

3. FRAMEWORK AND PROBLEM DEFINITION

In this section we present the essential definitions, which are later used to define our ranking metrics. We also formally define the keyword search problem and outline the ranking factors.

3.1 Database Graph, Schema, and Authority Transfer Graph

We view a database as a labeled graph, which is a model that easily captures both relational and XML databases. The *data graph* $D(V_D, E_D)$ is a labeled directed graph where every node v has a label $\lambda(v)$ and a set of keywords. For example, the node "ICDE 1997" of Figure 1 has label "Year" and the set of keywords $\{$`''ICDE''`, `''1997''`, `''Birmingham''`$\}$. Each node represents an *object* of the database and may have a sub-structure. Without loss of generality, ObjectRank assumes that each node has a tuple of attribute name/attribute value pairs. For example, the "Year" nodes of Figure 1 have `name`, `year` and `location` attributes. Notice that the keywords appearing in the attribute values comprise the set of keywords associated with the node. One may assume richer semantics by including the metadata of a node in the set of keywords. For example, the metadata "Forum", "Year", "Location" could be included in the keywords of a node. The specifics of modeling the data of a node are orthogonal to ObjectRank and will be neglected in the rest of the discussion.

Each edge e from u to v is labeled with its *role* $\lambda(e)$ (we overload λ) and represents a relationship between u and v. For example, every "paper" to "paper" edge of Figure 1 has the label "cites". When the role is evident and uniquely defined from the labels of u and v, we omit the edge label. For simplicity we will assume that there are no parallel edges and we will often denote an edge e from u to v as "$u \rightarrow v$".

A critical issue in constructing the data graph for a database is to decide the granularity of the information in the nodes. For example, if we are to return a paper, should we also

Data Graph	*Nodes*	*Edges*
Relational Database	Tuples (or attribute values)	Primary-to-Foreign Key Relationships
XML Database	XML Elements (or XML Nodes)	Containment or ID-IDREF Edges
Web	Pages	Hyperlinks

Table I. Mapping of Common Data Models to a Data Graph.

return the author names and the conference where the paper was published? We adopt the idea of predefined "answer nodes" as described in [Bhalotia et al. 2002; Dar et al. 1998; Guo et al. 2003; Hristidis et al. 2003][1]. Hence, in the above example, we choose to store the author and conference information in every paper node. Keep in mind that the data graph is a conceptual structure, so the actual physical storage may vary.

The data graph can represent relational [Agrawal et al. 2002; Hristidis and Papakonstantinou 2002] and XML [Hristidis et al. 2003; Guo et al. 2003] databases, as well as the Web [Brin and Page 1998]. The mappings of these data models to nodes and edges of the data graph are shown in Table I.

The use of our ranking metrics does not require the existence of a schema. However, if a schema is present then it can be used to easier define the authority transfer rates (see below). Furthermore, the schema may offer optimization opportunities as discussed in Section 6. The *schema graph* $G(V_G, E_G)$ (Figure 4) is a directed graph that describes the structure of D. Every node has an associated label. Each edge is labeled with a role, which may be omitted, as discussed above for data graph edge labels. We say that a data graph $D(V_D, E_D)$ *conforms* to a schema graph $G(V_G, E_G)$ if there is a unique assingment μ of data-graph nodes to schema-graph nodes and a consistent assignment of edges such that:

(1) for every node $v \in V_D$ there is a node $\mu(v) \in V_G$ such that $\lambda(v) = \lambda(\mu(v))$;

(2) for every edge $e \in E_D$ from node u to node v there is an edge $\mu(e) \in E_G$ that goes from $\mu(u)$ to $\mu(v)$ and $\lambda(e) = \lambda(\mu(e))$.

Authority Transfer Schema Graph. From the schema graph $G(V_G, E_G)$, we create the *authority transfer schema graph* $G^A(V_G, E^A)$ to reflect the authority flow through the edges of the graph. This may be either a trial and error process, until we are satisfied with the quality of the results, or a domain expert's task. In particular, for each edge $e_G = (u \to v)$ of E_G, two *authority transfer edges*, $e_G^f = (u \to v)$ and $e_G^b = (v \to u)$ are created. The two edges carry the label of the schema graph edge and, in addition, each one is annotated with a (potentially different) *authority transfer rate* - $\alpha(e_G^f)$ and $\alpha(e_G^b)$ correspondingly. We say that a data graph conforms to an authority transfer schema graph if it conforms to the corresponding schema graph. (Notice that the authority transfer schema graph has all the information of the original schema graph.)

Figure 5 shows the authority transfer schema graph that corresponds to the schema graph of Figure 4 (the edge labels are omitted). The motivation for defining two edges for each edge of the schema graph is that authority potentially flows in both directions and not only in the direction that appears in the schema. For example, a paper passes its authority to its authors and vice versa. Notice however, that the authority flow in each direction (defined by the authority transfer rate) may not be the same. For example, a paper that is cited by

[1] In XKeyword [Hristidis et al. 2003] they are referred to as target objects.

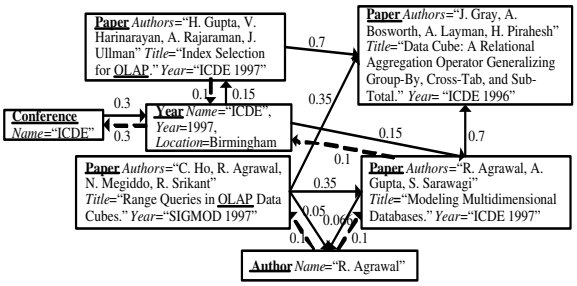

Fig. 6. Authority transfer data graph

important papers is clearly important but citing important papers does not make a paper important.

Notice that the sum of authority transfer rates of the outgoing edges of a schema node u may be less than 1[2], if the administrator believes that the edges starting from u do not transfer much authority. For example, in Figure 5, conferences only transfer 30% of their authority.

Authority Transfer Data Graph. Given a data graph $D(V_D, E_D)$ that conforms to an authority transfer schema graph $G^A(V_G, E^A)$, ObjectRank derives an *authority transfer data graph* $D^A(V_D, E_D^A)$ (Figure 6) as follows. For every edge $e = (u \to v) \in E_D$ the authority transfer data graph has two edges $e^f = (u \to v)$ and $e^b = (v \to u)$. The edges e^f and e^b are annotated with authority transfer rates $\alpha(e^f)$ and $\alpha(e^b)$. Assuming that e^f is of type e_G^f, then

$$\alpha(e^f) = \begin{cases} \frac{\alpha(e_G^f)}{\textit{OutDeg}(u, e_G^f)}, \ if \ \textit{OutDeg}(u, e_G^f) > 0 \\ 0, \ if \ \textit{OutDeg}(u, e_G^f) = 0 \end{cases} \tag{4}$$

where $\textit{OutDeg}(u, e_G^f)$ is the number of outgoing edges from u, of type e_G^f. The authority transfer rate $\alpha(e^b)$ is defined similarly. Figure 6 illustrates the authority transfer data graph that corresponds to the data graph of Figure 1 and the authority schema transfer graph of Figure 5. Notice that the sum of authority transfer rates of the outgoing edges of a node u of type $\mu(u)$ may be less than the sum of authority transfer rates of the outgoing edges of $\mu(u)$ in the authority transfer schema graph, if u does not have all types of outgoing edges.

3.2 Keyword Search and Ranking Factors

A *keyword query* q is defined as a set of keywords. The result of a keyword query is a list of objects of the database (i.e., nodes of the data graph), ranked according to the query. The ranking is performed according to three desired properties presented below. We explain how our system measures each of these properties by exploiting the link-structure of the data graph. Notice that there is other non-link-based factors (e.g., IR score of individual nodes [Hristidis et al. 2003]) that can be incorporated in the ranking as well, but they are beyond the scope of this paper.

Relevance to Query: ObjectRank We should rank higher results that either contain the

[2]In terms of the random walk model, this would be equivalent to the disappearance of a surfer.

keywords of the query or are semantically associated to the keywords of the query. The latter factor is equivalent to being connected through paths on the data graph in our data model, where edges correspond to semantic associations. In our system, the link-based relevance of a node v to a query w (assume a single-keyword query for now) is the ObjectRank value $r^w(v)$ of v discussed in Section 4.1.

Specificity: Inverse ObjectRank Specific results (nodes) should be ranked higher. That is, results with content particular to the query are preferred over results with content that spans across many topics. Previous work has not considered any link-based specificity metric. In Section 4.2 we present and discuss in detail Inverse ObjectRank.

Global quality: Global ObjectRank Results of high quality should be ranked higher. The link-structure of the data graph is used to measure quality. In particular, nodes with high incoming authority flow are assumed to have higher quality [Brin and Page 1998; Guo et al. 2003]. For example, a highly referenced paper should be ranked higher than a non-referenced paper if the other ranking properties are equal. In our system, we use Global ObjectRank (defined in Section 4.1), which is an effective link-based metric to measure the global authority, that is, the quality of a node of the data graph. The Global ObjectRank $r^G(u)$ of a node u is defined as the probability that a random surfer starting from any node of the authority transfer graph will be at u at a specific time. For the case of the Web, Global ObjectRank is equivalent to PageRank [Brin and Page 1998], whose value has been proven by the success of Google[3].

Notice that these three properties correspond to the specificity, keyword proximity and hyperlink awareness properties respectively, defined in XRANK [Guo et al. 2003]. The same three properties (although not explicitly enumerated) have been used in other works as well (e.g., [Bhalotia et al. 2002]).

4. OBJECTRANK AND INVERSE OBJECTRANK

In this section we present the ranking metrics we use: ObjectRank, Global ObjectRank and Inverse ObjectRank. Furthermore, we explain the parallelisms to Information Theory (Section 4.3) metrics. Finally, in Section 4.4 we present and address the challenges in combining these metrics into a ranking function.

4.1 ObjectRank

We first define ObjectRank for a single keyword. In Section 4.4 we extend to multiple keywords. Given a single keyword query w, ObjectRank finds the *keyword base set* $S(w)$ (from now on referred to simply as base set when the keyword is implied) of objects that contain the keyword w and assigns an ObjectRank $r^w(v_i)$ to every node $v_i \in V_D$ by resolving the equation

$$\mathbf{r}^w = d\mathbf{A}\mathbf{r}^w + \frac{(1-d)}{|S(w)|}\mathbf{s} \tag{5}$$

where $A_{ij} = \alpha(e)$ if there is an edge $e = (v_j \rightarrow v_i)$ in E_D^A and 0 otherwise, d controls the base set importance, and $\mathbf{s} = [s_1, \ldots, s_n]^T$ is the base set vector for $S(w)$, i.e., $s_i = 1$ if $v_i \in S(w)$ and $s_i = 0$ otherwise.

[3]http://www.google.com

The damping factor d determines the portion of ObjectRank that an object transfers to its neighbors as opposed to making a random jump to one of the base set pages. It was first introduced in the original PageRank paper [Brin and Page 1998], where it was used to ensure convergence in the case of PageRank sinks. However, in addition to that, in our work it is a calibrating factor, since by decreasing d, we favor objects that actually contain the keywords (i.e., are in base set)as opposed to objects that acquire ObjectRank through the incoming edges. The value for d used by PageRank [Brin and Page 1998] is 0.85, which we also adopt when we want to balance the importance of containing the actual keywords as opposed to being pointed by nodes containing the keywords.

Global ObjectRank. The definition of global ObjectRank is different for different applications or even users of the same application. In this work, we focus on cases where the global ObjectRank is calculated applying the random surfer model, and including all nodes in the base set. The same calibrating parameters are available, as in the keyword-specific ObjectRank. Notice that this way of calculating the global ObjectRank, which is similar to the PageRank approach [Brin and Page 1998], assumes that all nodes (pages in PageRank) initially have the same value. However, there are many applications where this is not true, as we discuss in Section 10.

4.2 Inverse ObjectRank

Before presenting the specifics of Inverse ObjectRank, we explain why the traditional IR specificity metrics are inadequate. In particular, IR metrics ignore the link-structure which makes them incomplete. For example, the document length (*dl*) metric cannot distinguish between objects (nodes) of approximately the same length, as is the case in our bibliographic database of paper titles and author names. Traditional IR specificity metrics are complementary to Inverse ObjectRank since they focus on the nodes of the authority flow graph, whereas Inverse ObjectRank exploits the edges. In this work we only evaluate Inverse ObjectRank and other alternative link-structure based specificity metrics in Section 7.2.

The intuition behind Inverse ObjectRank is the following. Given a keyword w, the ObjectRank value of a node v is the probability that a random surfer starting from a node containing w will be at v at a specific time. v is *specific* with respect to w if there is only few such keywords for which a surfer will end up on v starting from them. That is, if the random surfer will start at v and follow the edges of the authority transfer graph on the reverse direction, he/she should land back on w with high probability.

The above intuition is formally defined as follows. We first need to define the *inverse authority transfer graph* $D^I(V_D, E_D^I)$, given the authority transfer data graph $D^A(V_D, E_D^A)$, as follows: For every edge $e(u \to v) \in E_D^A$, we create an opposite-direction edge $e^I(v \to u) \in E_D^I$ with authority flow rate $a(e^I) = a(e)\frac{OutDeg(u)}{InDeg(v)}$. Notice that $1/OutDeg(u)$ is used in the calculation of $a(e)$, so by multiplying by $OutDeg(u)$ this is evened out.

Given a single-keyword query $q = \{w\}$, the Inverse ObjectRank score $p^w(u)$ of a node u is the probability that a random surfer of the inverse authority transfer graph D^I starting from u will be at a node containing w at a specific time.

Inverse ObjectRank is calculated in two steps. First, for each node $v \in D^I$ we compute its *connectivity*[4] $q^u(v)$ to u, i.e., how much authority starting from u will reach v through

[4]This could also be called Inverse ObjectRank with respect to u. However, we avoid using this name which we

D^I.

$$\mathbf{q}^u = d\mathbf{A}^I\mathbf{q}^u + (1-d)\mathbf{s}_u \tag{6}$$

where A^I is the transition matrix of D^I. That is, $A^I_{ij} = \alpha(e)$ if there is an edge $e = (v_j \rightarrow v_i)$ in D^I and 0 otherwise. $\mathbf{s}_u = [s_{u1}, \ldots, s_{un}]^T$ is the base set vector containing just u, i.e., $s_{ui} = 1$ if v_i is u and $s_{ui} = 0$ otherwise. Note that the connectivity $q^u(v)$ of a node v is equivalent to the inverse P-distance from u to v as defined by Jeh and Widom [Jeh and Widom 2003].

Second, the Inverse ObjectRank $p^w(u)$ is computed by summing the connectivities $q^u(v)$ of all nodes that contain w. That is

$$p^w(u) = \sum_{v \in S(w)} q^u(v) \tag{7}$$

where $S(w)$ is the base set of w as defined in Equation 5.

Global Inverse ObjectRank $\mathbf{p}$, which we do not use in our ranking function but has its own merit, is calculated by Equation 8. High Global Inverse ObjectRank denotes high connectivity of a node in a way similar to hub nodes in [Kleinberg 1999].

$$\mathbf{p} = d\mathbf{A}^I\mathbf{p} + \frac{1-d}{|V|}\mathbf{e} \tag{8}$$

where $\mathbf{e} = [1, \ldots, 1]^T$.

Notice that Inverse ObjectRank is a keyword-specific metric of specificity, in the same sense that ObjectRank is a keyword-specific metric of relevance. This is the key reason why it performs superior to keyword-independent specificity heuristic metrics (including Global Inverse ObjectRank) as we show in Section 7. Also notice that Inverse ObjectRank has the same convergence properties as ObjectRank, which are described in Section 2.

4.3 Information Theory Perspective

In this section we discuss Inverse ObjectRank from an Information Theory perspective. In particular, we show how the link-based factors described in Section 3.2 appear in the context of Information Theory formulas. In general, the ranking functions in Information Retrieval can be explained as the increase of information when specifying a term w_i [Aizawa 2000]. In particular, the famous $tf \cdot idf$ ranking function can be explained using this approach [Aizawa 2000]. We apply the same Information Theory principle to create a ranking formula for graph databases as follows.

Let V and W be the sets of nodes (documents in IR) and keywords in the database. The information increase of V after the event of observing w_i can be expressed using the Kullback-Leibler information metric, which is a measure of the difference between two probability distributions. Kullback-Leibler information between $P(V|w_i)$ and $P(V)$, where $P(.)$ denotes probability, is calculated by

$$K(P(V|w_i), P(V)) = \sum_{v_j \in V} log\frac{P(v_j|w_i)}{P(v_j)}. \tag{9}$$

reserve for the product of the final (second) step of the computation.

Using Bayes rule, this can also be written as

$$K(P(V|w_i), P(V)) = \sum_{v_j \in V} log \frac{P(w_i|v_j)}{P(w_i)}. \quad (10)$$

In Information Retrieval $P(v_j|w_i)$ is the probability that document v_j contains keyword w_i and $P(v_j)$ is the probability of v_j, which is the same for all documents, that is, $P(v_j) = 1/n$, where n is the total number of documents. On the other hand, the equivalent quantity in a graph database is the probability that starting from a node containing w_i, a random surfer will be at node v_j at a specific time, that is, $P(v_j|w_i) = r^{w_i}(v_j)$. Similarly $P(w_i|v_j) = p^{w_i}(v_j)$. Also, $P(v_j)$ is the Global ObjectRank value of v_j, that is, $P(v_j) = r^G(v_j)$. Finally $P(w_i)$ is common for all nodes since it is only query dependent and can hence be ignored.

Depending on whether we adopt Equation 9, Equation 10 or a combination of the two, we generate ranking functions that use ObjectRank for relevance, and Global ObjectRank or Inverse ObjectRank for specificity. In Section 7, we qualitatively compare these combinations.

4.4 Combine Ranking Factors and Multiple Keywords

There are two levels of combining scores in our framework to reach a ranking function for node v given a multiple-keyword query "$q = \{w_1, \ldots, w_m\}$". First, we need to find the score $f^{w_i}(v)$ ($f^{w_i}(v)$ is the score of node v given keyword w_i) of v for every single keyword w_i, and then combine these scores (and possibly Global ObjectRank $r^G(v)$) to compute the final score $f^q(v)$.

First, we define two alternative ways to combine ObjectRank with Inverse ObjectRank to compute $f^{w_i}(v)$, shown in Equations 11 and 12. The two equations are used to boost or downplay the weight of Inverse ObjectRank, that is, of the specificity factor in a keyword query respectively.

$$f^{w_i}(v) = r^{w_i}(v) \cdot p^{w_i}(v) \quad (11)$$

$$f^{w_i}(v) = r^{w_i}(v) \cdot \sqrt{p^{w_i}(v)} \quad (12)$$

Alternatively, if we choose a different specificity metric (see Section 7) we can replace $p^{w_i}(v)$ in Equation 11 by that metric, where we also show that Equation 12 typically produces superior results.

Second, we define the semantics of a multiple-keywords query "$q = \{w_1, \ldots, w_m\}$" by naturally extending the multiple-keywords random walk model. In particular, for the case of ObjectRank we consider m independent random surfers, where the ith surfer starts from the keyword base set $S(w_i)$. For AND semantics, the ObjectRank of an object v with respect to the m-keywords query is the probability that, at a given point in time, the m random surfers are simultaneously at v. We extend this model by substituting $r^{w_i}(v)$ by $f^{w_i}(v)$. Hence the score $f^q(v)$ of node v with respect to the m keywords is

$$f^{w_1,\ldots,w_m}(v) = \prod_{i=1,\ldots,m} f^{w_i}(v). \quad (13)$$

For OR semantics, the ObjectRank of v is the probability that, at a given point in time, *at least one* of the m random surfers will reach v. Hence, for two keywords w_1 and w_2 the model can be extended to

(a)

47.31	11.44	An XML Indexing Structure with Relative Region Coordinate. Dao Dinh Kha, ICDE 2001
41.02	3.08	DataGuides: Enabling Query ... Optimization in Semistructured... Roy Goldman, VLDB 1997
7.44	28.43	Access Path Selection in a RDBMS. Patricia G. Selinger, SIGMOD 1979
31.44	3.24	Querying Object-Oriented Databases. Michael Kifer, SIGMOD 1992
26.73	3.09	A Query … Optimization Techniques for Unstructured Data. Peter Buneman, SIGMOD 1996

(b)

47.31	11.44	An XML Indexing Structure with Relative Region Coordinate. Dao Dinh Kha, ICDE 2001
7.44	28.43	Access Path Selection in a RDBMS. Patricia G. Selinger, SIGMOD 1979
2.04	102.1	R-Trees: A Dynamic Index Structure for Spatial Searching. Antonin Guttman, SIGMOD 1984
1.73	112.7	The K-D-B-Tree: A Search Structure For Large … Indexes. John T. Robinson, SIGMOD 1981
41.02	3.08	DataGuides: Enabling Query … Optimization in Semistructured... Roy Goldman, VLDB 1997

Fig. 7. Top 5 papers on "XML Index", with and without emphasis on "XML"

$$f^{w_1,w_2}(v) = f^{w_1}(v) + f^{w_2}(v) - f^{w_1}(v)f^{w_2}(v) \tag{14}$$

and for more than two it is defined accordingly, as specified by the inclusion-exclusion principle (also known as the sieve principle). Notice that [Haveliwala 2002] also takes the sum of the topic-sensitive PageRank values to calculate the PageRank of a page.

If Global ObjectRank is included in the computation, it is treated as an additional keyword w_{m+1} with $f^{w_{m+1}}(v) = r^G(v)$.

Weigh keywords by frequency. A drawback of the *combining function* of Equation 13 is that it favors the more popular keywords in the query. The reason is that the distribution of ObjectRank values is more skewed when the size $|S(w)|$ of the base set $S(w)$ increases, because the top objects tend to receive more references. For example, consider two results for the query "XML AND Index" shown in Figure 7. Result (b) corresponds to the model described above. It noticeably favors the "Index" keyword over the "XML". The first paper is the only one in the database that contains both keywords in the title. However, the next three results are all classic works on indexing and do not apply directly to XML. Intuitively, "XML" as a more specific keyword is more important to the user. Indeed, the result of Figure 7 (a) was overwhelmingly preferred over the result of Figure 7 (b) by participants of our relevance feedback survey (Section 7). The latter result contains important works on indexing in semistructured, unstructured, and object-oriented databases, which are more relevant to indexing of XML data. This result is obtained by using the modified formula:

$$r^{w_1,\ldots,w_m}(v) = \prod_{i=1,\ldots,m} (r^{w_i}(v))^{g(w_i)} \tag{15}$$

where $g(w_i)$ is a *normalizing exponent*, set to $g(w_i) = 1/log(|S(w_i)|)$. This exponent plays a role similar to the inverse document frequency (idf) in traditional Information Retrieval. Using the normalizing exponents g("XML") and g("Index") in the above example is equivalent to running in parallel g("XML") and g("Index") random walks for the "XML" and the "Index" keywords respectively.

Compare to single base set approach. One can imagine alternative semantics to calculate the ObjectRank for multiple keywords, other than combining the single-keyword ObjectRanks. In particular, consider combining all objects with at least one of the keywords into a single base set. Then a single execution of the ObjectRank algorithm is used to determine the scores of the objects. Incidentally, these semantics were used in the HITS system [Kleinberg 1999]. We show that such "single base set" semantics can be achieved by combining single-keyword ObjectRank values applying appropriate exponents. Furthermore, we explain how our semantics avoid certain problems of "single base set" semantics.

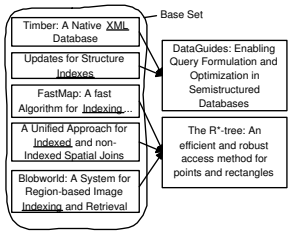

Fig. 8. Example where "HITS" approach fails in AND semantics.

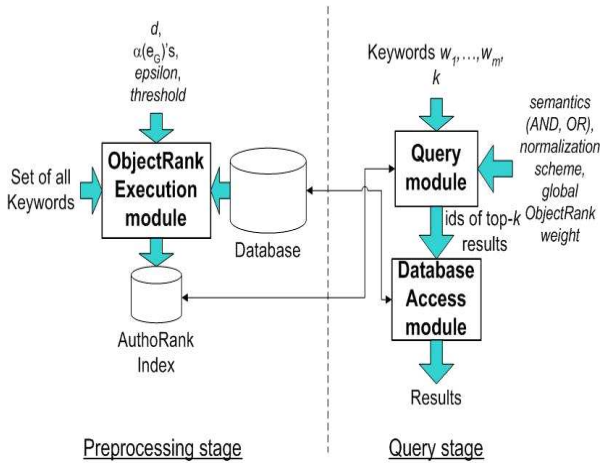

Fig. 9. System Architecture.

In order to compare to the "single base set" approach for AND semantics (Equation 13), we consider two scenarios and assume without loss of generality that there are two keywords. First, assume that we only put in the base set S objects that contain both keywords. These objects will be in both keyword-specific base sets as well, so these objects and objects pointed by them will receive a top rank in both approaches. Second, if S contains objects containing any of the two keywords, we may end up ranking highest an object that is only pointed by objects containing one keyword. This cannot happen with the keyword-specific base sets approach. For example, in Figure 8, the "single base set" approach would rank the R^* paper higher than the DataGuides paper for the query "XML AND Index", even though the R^* paper is irrelevant to XML.

For OR semantics (Equation 14), the base set S in the"single base set" approach is the union of the keyword-specific base sets. We compare to an improved version of the "single base set" approach, where objects in base set are weighted according to the keywords they contain, such that infrequent keywords are assigned higher weight. In particular, if an object contains both keywords, for a two keyword query, it is assigned a base ObjectRank of $(1-d)\cdot(\frac{1}{|S(w_1)|}+\frac{1}{|S(w_2)|})$. Then, using the Linearity Theorem in [Jeh and Widom 2003], we can prove that the ObjectRanks calculated by both approaches are the same.

5. ARCHITECTURE

We have implemented a system to answer keyword queries on databases. The user inputs (a) a set of keywords, (b) a choice for combining semantics (AND or OR), (c) the importance of global quality of the results (i.e., Global ObjectRank), (d) the importance of containing the actual query keywords (translated to a damping factor value d), and (e) a specificity metric (as we explain in Section 7). The output of the system is a ranked list of nodes of the database (to be more formal, of the authority transfer graph) according to the input parameters based on the ranking function in Equation 13 or 14 (for AND and OR semantics respectively). The authority transfer graph is stored in a relational database using the schema shown in Figure 4.

The architecture of the system, which is shown in Figure 9, is divided into two stages. The preprocessing stage consists of the *Authority Flow Execution module*, which inputs the authority transfer graph G to be indexed, the set of all keywords that will be indexed, and a set of parameters. In particular these parameters are: (i) A set of damping factors d, that users are expected to choose from. (ii) The convergence constant $epsilon$ which determines when the ObjectRank and Inverse ObjectRank algorithms converge, and (iii) The $threshold$ value which determines the minimum score that an object must have to be stored in the authority flow index. Note that other index pruning techniques are possible [Carmel et al. 2001]; however, we found that this simple uniform pruning technique performs well in our setting.

The Authority Flow Execution module creates the *authority flow index*, which is an inverted index, indexed by the keywords. For each keyword w, it stores a list of $\langle id(u), f^w(u)\rangle$ pairs for each object u that has $f^w(u) \geq threshold$. The pairs are sorted by descending $f^w(u)$ to facilitate an efficient querying method as we describe below. The authority flow index has been implemented as an index-based table, where the lists are stored in a CLOB attribute. A hash-index is built on top of each list to allow random access, which is required by the Query module. Note that if we allow multiple combinations of calibration parameters to be selected by the user, then we create multiple inverted indexes, one for each such combination.

The *Query module* inputs a set of keywords $w_1, \ldots, w_m$ and a set of adjusting parameters, and outputs the top-k objects according to the ranking function (Equation 13 or 14). In particular, these parameters are: (a) a choice for combining semantics (AND or OR), (b) the importance of global quality of the results (i.e., Global ObjectRank), (c) the importance of containing the actual query keywords (translated to a damping factor value d), and (d) a specificity metric (as we explain in Section 7). The keyword-specific lists read from the authority flow index are merged using the *Threshold Algorithm* [Fagin et al. 2001] which is guaranteed to read the minimum prefix of each list. Notice that the Threshold Algorithm is applicable since both combining functions (Equations 13 and 14) are monotone.

Finally, the *Database Access module* inputs the result ids and queries the database to get the corresponding node of the authority transfer graph. This information is stored into an id-indexed table, that contains a CLOB attribute value for each object id. For example, a paper object CLOB would contain the paper title, the authors' names, and the conference name and year.

5.1 Demo

We have built a demo [Hwang et al. 2006] on bibliographic data, which is available online at two mirror sites: http://www.db.ucsd.edu/ObjectRank and http://dbir.cis.fiu.edu/BibObjectRank. The data was collected using the following method. First, we downloaded all publications and citations from the DBLP database[5]. We noticed that this source is missing too many citations, which greatly degrades the quality of link-based analysis. To overcome this shortcoming, we used Citeseer[6] as an additional citations' source. We built a web crawler to retrieve these citations since we found that the exported files of Citeseer are in a large degree inaccurate. We matched papers from the two sources using their titles, which of course can lead to few inaccurate matches.

Our demo offers to the user multiple authority flow settings, in order to accommodate multiple user profiles/requirements. We believe the ability to customize authority flow schemes is important, since we should not assume that "one size fits all" when it comes to opinions about authority flow. For example, there is one setting for users that primarily care for papers with high global importance and another for users that primarily care for papers that are directly or indirectly heavily referenced by papers that have the keywords. We expect that multiple settings make sense in all non-trivial applications.

6. INDEX CREATION ALGORITHMS

This section presents algorithms to create the ObjectRank index, which can be adjusted to compute Inverse ObjectRank as well. Section 6.1 presents an algorithm for the case of arbitrary authority transfer data graphs D^A. Sections 6.2 and 6.3 show how we can do better when D^A is a directed acyclic graph (DAG) and "almost" a DAG respectively (the latter property is explained in Section 6.3). Sections 6.4 and 6.5 present optimizations when the authority transfer graph has a small vertex cover, or is a DAG of subgraphs respectively. Finally, Section 6.6 presents optimization opportunities based on manipulating the initial values of the iterative algorithm.

6.1 General algorithm

Figure 10 shows the algorithm that creates the ObjectRank Index. The algorithm accesses the authority transfer data graph D^A many times, which may lead to a too long execution time if D^A is very large. Notice that this is usually not a problem, since D^A only stores object ids and a set of edges which is small enough to fit into main memory for most databases. Notice that lines 2-4 correspond to the original PageRank calculation [Brin and Page 1998] modulo the authority transfer rates information.

6.2 DAG algorithm

There are many applications where the authority transfer data graph is a DAG. For example a database of papers and their citations (ignoring author and conference objects), where each paper only cites previously published papers, is a DAG. Figure 11 shows an improved algorithm, which makes a single pass of the graph D^A and computes the actual ObjectRank values. Notice that there is no need for $epsilon$ any more since we derive the

[5]http://www.informatik.uni-trier.de/ ley/db/
[6]http://citeseer.ist.psu.edu/

```
CreateIndex(keywordsList, epsilon, threshold, α(.), d){
01. For each keyword w in keywordsList do {
02.   While not converged do
03.   /*i.e., ∃v, |r^(k+1)(v) − r^(k)(v)| > epsilon*/
04.     MakeOnePass(w,α(.), d);
05.   StoreObjectRanks();
06. }
}
MakeOnePass(w,α(.), d) {
07. Evaluate Equation 5 using the r from the previous iteration on the right side;
}
StoreObjectRanks() {
08. Sort the ⟨id(i), r(v_i)⟩ pairs list by r(v_i) and store it in inverted index, after removing pairs with r(v_i) < threshold;
}
```

Fig. 10. Algorithm to create ObjectRank Index

precise solution of Equation 5, in contrast to the algorithm of Figure 10 which calculates approximate values. The intuition is that ObjectRank is only transferred in the direction of the topological ordering, so a single pass suffices. Notice that topologically sorting a graph $G(V, E)$ takes time $\Theta(V + E)$ [Cormen et al. 1989] in the general case. In many cases the semantics of the database can lead to a better algorithm. For example, in the papers database, we can efficiently topologically sort the papers by first sorting the conferences by date. This method is applicable for databases where a temporal or other kind of ordering is implied by the link structure.

```
CreateIndexDAG(keywordsList, threshold, α(.), d){
01. Topologically sort nodes in graph D^A;
02. /*Consecutive accesses to D'^A are in topological order.*/
03. For each keyword w in keywordsList do {
04.   MakeOnePass(w,α(.), d);
05.   StoreObjectRanks();
06. }
}
```

Fig. 11. Algorithm to create ObjectRank Index for DAGs

In the above example, the DAG property was implied by the semantics. However, in some cases we can infer this property by the structure of the authority transfer schema graph G^A, as the following theorem shows.

THEOREM 6.1. *The authority transfer data graph D^A is a DAG if and only if*

—the authority transfer schema graph G^A is a DAG, or

—for every cycle c in G^A, the subgraph D'^A of D^A consisting of the nodes (and the edges connecting them), whose type is one of the schema nodes of c, is a DAG.

6.3 Almost-DAG algorithm

The most practically interesting case is when the authority transfer data graph D^A is *almost* a DAG, that is, there is a "small" set U of *backedges*, and if these edges are removed, D^A becomes a DAG. Notice that the set U is not unique, that is, there can be many *minimal* (i.e., no edge can be removed from U) sets of backedges. Instead of working with the set of backedges U, we work with the set L of *backnodes*, that is, nodes from which the backedges start. This reduces the number of needed variables as we show below, since $|L| \leq |U|$.

In the papers database example (when author and conference objects are ignored), L is the set of papers citing a paper that was not published previously. Similarly, in the complaints database (Figure 14), most complaints reference previous complaints. Identifying the minimum set of backnodes is NP-complete[7] in the general case. However, the semantics of the database can lead to efficient algorithms. For example, for the databases we discuss in this paper (i.e, the papers and the complaints databases), a backnode is simply an object referencing an object with a newer timestamp.

The intuition of the algorithm (Figure 12) is as follows: the ObjectRank of each node can be split to the DAG-ObjectRank which is calculated ignoring the backedges, and the backedges-ObjectRank which is due to the backedges.

To calculate backedges-ObjectRank we assign a variable c_i to each backnode c_i (for brevity, we use the same symbol to denote a backnode and its ObjectRank), denoting its ObjectRank. Before doing any keyword-specific calculation, we calculate how c_i's are propagated to the rest of the graph D^A (line 5), and store this information in $\mathbf{C}$. Hence C_{ij} is the coefficient with which to multiply c_j when calculating the ObjectRank of node v_i. To calculate $\mathbf{C}$ (lines 13-15) we assume that the backedges are the only source of ObjectRank, and make one pass of the DAG in topological order.

Then, for each keyword-specific base set: (a) we calculate the DAG-ObjectRanks $\mathbf{r}'$ (line 7) ignoring the backedges (but taking them into account when calculating the outgoing degrees), (b) calculate c_i's solving a linear system (line 8), and (c) calculate the total ObjectRanks (line 10) by adding the backedge-ObjectRank ($\mathbf{C} \cdot \mathbf{c}$) and the DAG-ObjectRank($\mathbf{r}'$). Each line of the system of line 8 corresponds to a backnode $c_i \equiv v_j$ (i.e., the ith backnode is the jth node of the topologically sorted authority transfer data graph D'^A), whose ObjectRank c_i is the sum of the backedge-ObjectRank ($\mathbf{C}_j \cdot \mathbf{c}$) and the DAG-ObjectRank ($\mathbf{r}'_j$). The overline notation on the matrices of this equation selects the L lines from each table that correspond to the backnodes. We further explain the algorithm using an example.

EXAMPLE 1. *The graph D^A is shown in Figure 13 (a). Assume $d = 0.5$ and all edges are of the same type t with authority transfer rate $\alpha(t) = 1$. First we identify the backnodes $c_1 \equiv P_5, c_2 \equiv P_4$ and then we topologically sort the graph ignoring the backedges corresponding to the backnodes, depicted with dotted arrows in Figure 13 (a). Then we create the coefficients table C (line 5), as follows:*

$$
\begin{aligned}
r(P_1) &= 0 \\
r(P_2) &= 0.5 \cdot 0.5 \cdot c_2 = 0.25 \cdot c_2 \\
r(P_3) &= 0.5 \cdot c_1
\end{aligned}
$$

[7] Proven by reducing Vertex Cover to it.

CreateIndexAlmostDAG(keywordsList, $threshold$, $\alpha(.)$, d){
01. $\mathbf{c}$: vector of ObjectRanks of backnodes;
02. Identify backnodes, and topologically sort the DAG (D^A without the backedges) D'^A;
03. /*Consecutive accesses to D'^A are in topological order.*/
04. /*Backedges are considered in D'^A for $\alpha(.)$.*/
05. $\mathbf{C}$=BuildCoefficientsTable();
06. For each keyword w in keywordsList do {
07. Calculate ObjectRanks vector $\mathbf{r}'$ for D'^A executing MakeOnePass(w,$\alpha(.)$, d);
08. Solve $\mathbf{c} = \overline{\mathbf{C}} \cdot \mathbf{c} + \overline{\mathbf{r}'}$;
09. /*$\overline{\mathbf{D}}$ denotes keeping only the lines of $\mathbf{D}$ corresponding to backnodes.*/
10. $\mathbf{r} = \mathbf{C} \cdot \mathbf{c} + \mathbf{r}'$
11. StoreObjectRanks();
12. }
}
BuildCoefficientsTable(){
13. For each node v_j do
14. $r(v_j) = d \cdot \sum_{backnode\ c_i\ points\ at\ v_j} (\alpha(c_i \rightarrow v_j) \cdot c_i) + d \cdot \sum_{non-backnode\ v_l\ points\ at\ v_j} (\alpha(v_l \rightarrow v_j) \cdot r(v_l))$;
15. Return $\mathbf{C}$, such that $\mathbf{r} = \mathbf{C} \cdot \mathbf{c}$
}

Fig. 12. Algorithm to create ObjectRank Index for *almost* DAGs

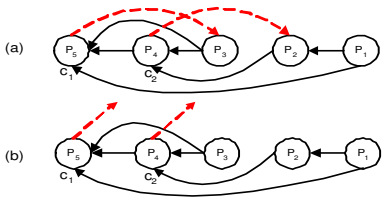

Fig. 13. Almost DAG.

$$r(P_4) = 0.5 \cdot r(P_2) + 0.5 \cdot 0.5 \cdot r(P_3) = 0.125 \cdot c_1 + 0.125 \cdot c_2$$
$$r(P_5) = 0.5 \cdot 0.5 \cdot r(P_3) + 0.5 \cdot 0.5 \cdot r(P_4) = 0.156 \cdot c_1 + 0.031 \cdot c_2$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 \\ 0 & 0.25 \\ 0.5 & 0 \\ 0.125 & 0.125 \\ 0.156 & 0.031 \end{bmatrix}$$

Assume we build the index for one keyword w contained in nodes P_1, P_3. We calculate (line 7) ObjectRanks for D'^A (taken by removing the backedges (dotted lines) from D^A).

$$r(P_1) = 0.5$$
$$r(P_2) = 0.5 \cdot 0.5 \cdot r(P_1) = 0.125$$
$$r(P_3) = 0.5$$
$$r(P_4) = 0.5 \cdot 0.5 \cdot r(P_3) + 0.5 \cdot r(P_2) = 0.188$$
$$r(P_5) = 0.5 \cdot 0.5 \cdot r(P_4) + 0.5 \cdot 0.5 \cdot r(P_3) + 0.5 \cdot 0.5 \cdot r(P_1) = 0.297$$

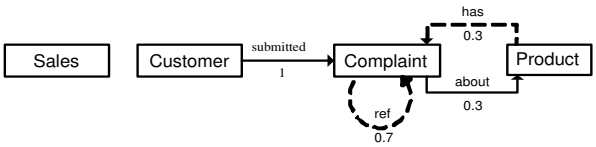

Fig. 14. Authority transfer schema graph for Complaints database.

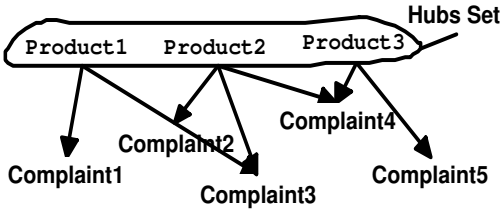

Fig. 15. Hierarchical-graph.

$\mathbf{r}' = [0.5\ 0.125\ 0.5\ 0.188\ 0.297]^T$

Solving the equation of line 8:

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0.156 & 0.031 \\ 0.125 & 0.125 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} 0.297 \\ 0.188 \end{bmatrix}$$

we get: $\mathbf{c} = [0.361\ 0.263]^T$*, where the overline-notation selects from the matrices the 5-th and the 4-th lines, which correspond to the backnodes* c_1 *and* c_2 *respectively. The final ObjectRanks are (line 10):* $\mathbf{r} = [0.5\ 0.190\ 0.680\ 0.266\ 0.361]^T$.

This algorithm can be viewed as a way to reduce the $n \times n$ ObjectRank calculation system of Equation 5, where n is the size of the graph, to the much smaller $|L| \times |L|$ equations system of line 8 of Figure 12. Interestingly, the two equations systems have the same format $\mathbf{r} = \mathbf{A}\mathbf{r} + \mathbf{b}$, only with different coefficient tables $\mathbf{A}, \mathbf{b}$. The degree of reduction achieved is inversely proportional to the number of backnodes.

The linear, first-degree equations system of line 8 can be solved using any of the well-studied arithmetic methods like Jacobi and Gauss-Seidel [Golub and Loan 1996], or even using the PageRank iterative approach which is simpler because we do not have to solve each equation with respect to a variable. The latter is shown to perform better in Section 8.

6.4 Algorithm for graphs with small vertex cover

Similarly to the almost-DAG case, we can reduce the ObjectRank calculation to a much smaller system (than the one of Equation 5) if authority transfer data graph D^A contains a relatively small vertex cover H. For example, consider a subset of the complaints database (Figure 14) consisting of the products and the complaints (without the reference edge to other complaints). Then H is the set of the products (Figure 15).[8] We call the nodes of H hub-nodes.

[8] A complaint can refer to more than one products.

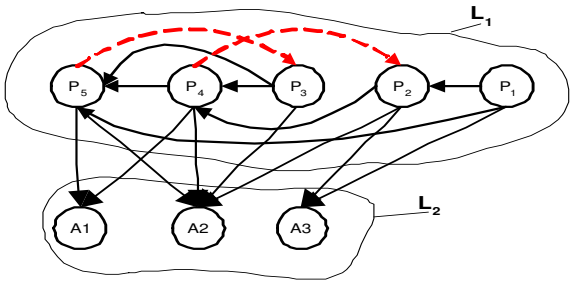

Fig. 16. Serializable Graph.

The intuition of the algorithm is the following: Let h_i be the ObjectRank of hub-node h_i. First, the ObjectRank of every non-hub-node i is expressed as a function of the ObjectRanks of the hub-nodes pointing to i. Then the ObjectRank of each hub-node h_i is expressed as a function of the non-hub-nodes pointing to h_i. This expression is equal to h_i, so we get $|H|$ such equations for the $|H|$ hub-nodes. Hence we reduce the computation to a $|H| \times |H|$ linear, first-degree system.

6.5 Serializing ObjectRank Calculation

This section shows when and how we can *serialize* the ObjectRank calculation of the whole graph $D^A(V_D, E_D^A)$ over ObjectRank calculations for disjoint, non-empty subsets $L_1, \ldots, L_r$ of V_D, where $L_1 \cup \ldots \cup L_r \equiv V_D$. The calculation is serializable if we first calculate the ObjectRanks for L_1, then use these ObjectRanks to calculate the ObjectRanks of L_2 and so on.

For example, consider the subset of the papers database consisting of the papers, their citations and the authors, where authority is transferred between the papers and from a paper to its authors (and not vice versa). Figure 16 shows how this authority transfer data graph can be serialized. In particular, we first calculate the ObjectRanks for the nodes in L_1 and then fo the nodes in L_2, as we elaborate below.

To define when the calculation is serializable, we first define the graph $D'^A(V', E')$ with $V' = \{L_1 \cup \ldots \cup L_r\}$ and $E' = \{(L_i, L_j) | \exists (v_i, v_j) \in E_D^A \wedge v_i \in L_i \wedge v_j \in L_j\}$. That is, there is an edge (L_i, L_j) in D'^A if there is an edge between two nodes $v_i \in L_i, v_j \in L_j$ of D^A. The following theorem defines when the ObjectRank calculation is serializable.

THEOREM 6.2. *The ObjectRank calculation for D^A is serializable iff D'^A is a DAG.*

The algorithm works as follows: Let $L_1, \ldots, L_r$ be topologically ordered. First, the ObjectRanks of the nodes in L_1 are computed ignoring the rest of D^A. Then we do the same for L_2, including in the computation the set I of nodes (and the corresponding connecting edges) of L_1 connected to nodes in L_2. Notice that the ObjectRanks of the nodes in I are not changed since there is no incoming edge from any node of L_2 to any node in I. Notice that any of the ObjectRank calculations methods described above can be used in each subset L_i.

6.6 Manipulating Initial ObjectRank values

All algorithms so far assume that we do a fresh execution of the algorithm for every keyword. However, intuitively we expect nodes with high Global ObjectRank to also have

high ObjectRank with respect to many keywords. We exploit this observation by assigning the Global ObjectRanks as initial values for each keyword specific calculation.

Furthermore, we investigate a space vs. time tradeoff. In particular, assume we have limitations on the index size. Then we only store a prefix (the first p nodes) of the nodes' list (recall that the lists are ordered by ObjectRank) for each keyword. During the query stage, we use these values as initial values for the p nodes and a constant (we experimentally found 0.03 to be the most efficient for our datasets) for the rest[9]. Both ideas are experimentally evaluated in Section 8.1.

7. QUALITATIVE EVALUATION

To evaluate the quality of the results to keyword queries we conducted a set of user surveys and we compared our results to a well-accepted ground truth source. Section 7.1 presents the results for variations of ObjectRank. Section 7.2 compares ways to express the specificity in the ranking function.

7.1 ObjectRank Evaluation

To evaluate the quality of the results of ObjectRank, we conducted two surveys. The first was performed on the DBLP database, with eight professors and Ph.D. students, who were not involved with the project. The second survey used the publications database of the IEEE Communications Society (COMSOC) [10] and involved five senior Ph.D. students from the Electrical Engineering Department.

Each participant was asked to compare and rank two to five lists of top-10 results for a set of keyword queries, assigning a score of 1 to 10, according to the relevance of the results list to the query. Each result list was generated by a different variation of the ObjectRank algorithm. One of the results lists in each set was generated by the "default" ObjectRank configuration which used the authority transfer schema graph of Figure 5 and $d = 0.85$. The users knew nothing about the algorithms that produced each result list. The survey was designed to investigate the quality of ObjectRank when compared to other approaches or when changing the adjusting parameters.

Effect of keyword-specific ranking. First, we assess the basic principle of ObjectRank, which is the keyword-specific scores. In particular, we compared the default (that is, with the parameters set to the values discussed in Section 1) ObjectRank with the global ranking algorithm that sorts objects that contain the keywords according to their global ObjectRank (where the base-set contains all nodes). Notice that this is equivalent to what Google used to[11] do for Web pages, modulo some minor difference on the calculation of the relevance score by Google. The DBLP survey included results for two keyword queries: "OLAP" and "XML". The score was 7:1 and 5:3 in favor of the keyword-specific ObjectRank for the first and second keyword query respectively. The COMSOC survey used the keywords "CDMA" and "UWB (ultra wideband)" and the scores were 4:1 and 5:0 in favor of the keyword-specific approach respectively.

Effect of authority transfer rates. We compared results of the default ObjectRank with

[9] Notice that, as we experimentally found, using the Global ObjectRanks instead of a constant for the rest nodes is less efficient. The reason is that if a node u is not in the top-p nodes for keyword k, u probably has a very small ObjectRank with respect to k. However u may have a great Global ObjectRank.

[10] http://www.comsoc.org

[11] Google's current ranking algorithm is not disclosed.

a simpler version of the algorithm that did not use different authority transfer rates for different edge types, i.e., all edge types were treated equally. In the DBLP survey, for both keyword queries, "OLAP" and "XML", the default ObjectRank won with scores 5:3 and 6.5:1.5 (the half point means that a user thought that both rankings were equally good) respectively. In the COMSOC survey, the scores for "CDMA" and "UWB" were 3.5:1.5 and 5:0 respectively.

Effect of the damping factor d. We tested three different values of the damping factor d: 0.1, 0.85, and 0.99, for the keyword queries "XML" and "XML AND Index" on the DBLP dataset. Two points were given to the first choice of a user and one point to the second. The scores were $2.5 : 8 : 13.5$ and $10.5 : 11.5 : 2$ (the sum is 24 since there are 8 users times 3 points per query) respectively for the three d values. We see that higher d values are preferred for the "XML", because "XML" is a very large area. In contrast, small d are preferable for "XML AND Index", because few papers are closely related to both keywords, and these papers typically contain both of them. The results were also mixed in the COMSOC survey. In particular, the damping factors 0.1, 0.85, and 0.99 received scores of 5:6:4 and 4.5:3.5:7 for the queries "CDMA" and "UWB" respectively.

Note that setting d to a very small value (e.g., $d = 0.1$ or less) is very close to using a traditional IR function like $tfidf$, because the majority of the authority stays in the nodes that contain the keywords. Furthermore the exponent in Equation 15 plays a role similar to idf. The tf metric also tends to be of minor importance in DBLP since words are rarely repeated in a title and almost never in an author name.

Effect of changing the weights of the keywords. We compared the combining functions for AND semantics of Equations 13 with the weighted combining method described in Section 4.4 for the two-keyword queries "XML AND Index" and "XML AND Query", in the DBLP survey. The use of the normalizing exponents proposed in Section 4.4 was preferred over the simple product function with ratios of 6:2 and 6.5:1.5 respectively. In the COMSOC survey, the same experiment was repeated for the keyword query "diversity combining". The use of normalizing exponents was preferred at a ratio of 3.5:1.5.

7.2 Inverse ObjectRank Evaluation

The user survey investigates and compares alternative ways to incorporate link-based specificity to keyword queries. In particular, we propose alternative specificity metrics and also experiment with various ways to incorporate Inverse ObjectRank in the ranking. We performed three qualitative experiments to compare these alternatives: a comparison to a textbook's bibliography, a user survey, and a quantitative measurement of the distances between the result lists. The key conclusion from these studies is that combining ObjectRank with the square root of Inverse ObjectRank produces the best results.

We consider the following ranking functions. For each case we specify the single keyword score $f^{w_i}(v)$ of node v as well as the multiple keywords combining function $f^{w_1,\ldots,w_m}(v)$. Notice that AND semantics is used.

(1) *Obj* ranks according to ObjectRank. $f^{w_i}(v) = r^{w_i}(v)$ and $f^{w_1,\ldots,w_m}(v)$ is defined by Equation 13.

(2) *ObjInv* ranks according to the product of ObjectRank and Inverse ObjectRank. $f^{w_i}(v)$ is defined by Equation 11 and $f^{w_1,\ldots,w_m}(v)$ by Equation 13.

(3) *ObjOverGlobal* uses the inverse of Global ObjectRank as the specificity metric. The

	Obj		*ObjInv*		*ObjOverGlobal*		*Objd03*		*ObjSqrtInv*	
	A-S	A-NS	A-S	A-NS	A-S	A-NS	A-S	A-NS	A-S	A-NS
tree index	7	1	6	1	0	0	6	1	7	1
hash index	3	3	1	0	0	0	0	0	2	1
concurrency control	4	2	7	0	0	0	7	1	7	1
object databases	1	4	3	0	0	0	4	2	4	1
deductive databases	4	2	4	0	0	0	4	0	5	0
spatial databases	3	2	1	0	0	0	2	0	2	0
distributed databases	1	3	5	0	0	0	5	1	6	1
relational model	3	5	3	2	0	0	3	2	3	4
query optimization	2	3	3	1	0	0	4	2	4	2
data mining	4	1	6	0	0	0	4	0	6	0
relational algebra	3	2	2	0	0	0	3	0	2	0
AVERAGE	3.18	2.55	3.73	0.36	0	0	3.82	0.82	4.36	1

Table II. Number of Authoritative-Specific and Authoritative-Non-Specific papers according to Textbook .

assumption is that if a node has high ObjectRank, it receives it from a wide range of nodes, and hence this node is too general. It is $f^{w_i}(v) = r^{w_i}(v)$ and $f^{w_1,\ldots,w_m}(v) = \prod_{i=1,\ldots,m} f^{w_i}(v)/r^G(v)$

(4) *Objd03* is the same as Obj but $d = 0.3$ ($d = 0.85$ when not specified). That is, this ranking attempts to achieve specificity by limiting the authority flow and emphasizing the nodes that contain the keywords.

(5) *ObjSqrtInv* ranks according to the product of ObjectRank and the square root of Inverse ObjectRank. $f^{w_i}(v)$ is defined by Equation 12 and $f^{w_1,\ldots,w_m}(v)$ by Equation 13.

(6) *ObjOverInc* uses the inverse of the number of incoming links $NumIncLinks(v)$ of node v as specificity metric. It is $f^{w_i}(v) = r^{w_i}(v)$ and $f^{w_1,\ldots,w_m}(v) = \prod_{i=1,\ldots,m} f^{w_i}(v)/NumIncLinks(v)$. $NumIncLinks(v)$ can be viewed as an approximation of $r^G(v)$, so this ranking can be viewed as an approximation of *ObjOverGlobal*.

(7) *ObjOverInvGlobal* uses the inverse of Global Inverse ObjectRank $r^{IG}(v)$ as the specificity metric. It is $f^{w_i}(v) = r^{w_i}(v)$ and $f^{w_1,\ldots,w_m}(v) = \prod_{i=1,\ldots,m} f^{w_i}(v)/r^{IG}(v)$.

Note that we do not compare to the document length (dl) which is the traditional IR specificity metric since all objects in DBLP have approximately the same length. *ObjOverInc* and *ObjOverInvGlobal* were found to perform much worse than the other ranking functions and their results are omitted for simplicity.

Compare to Textbook's Bibliography We assume that the bibliography section of each chapter in [Ramakrishnan and Gehrke 2003] is a highly credible source of references related to the chapter title. Based on this assumption, we compare the recall (precision is the same as recall in this case) of the top-10 papers produced by the five above ranking functions with respect to the papers in the bibliography section of the corresponding chapter, which is viewed as the ground truth.

We evaluated 11 queries that correspond to chapter titles of the textbook [Ramakrishnan and Gehrke 2003]. For each keyword query q, let $B(q)$ denote the set of papers in the bibliography of the corresponding chapter and $U(q)$ denote the set of papers that are in the bibliography of the book but not of that chapter, that is, they are not in $B(q)$. We assume that papers in $B(q)$ satisfy all properties of Section 3.2, that is, they are specific to q, relevant to q and of high quality. We refer to such papers as authoritative-specific

Obj	*ObjInv*	*ObjOverGlobal*	*Objd03*	*ObjSqrtInv*
2.13	3.42	2.13	3.60	3.92

Table III. Average Ratings of the Five Specificity Metrics at the User Survey.

for q. On the other hand, papers in $U(q)$ have high quality but are not highly relevant or specific to q, and are referred to as authoritative-non-specific. Table II shows the number of authoritative-specific and authoritative-non-specific papers for each query for the five ranking functions.

Obviously, *ObjOverGlobal* has the worst performance according to Table II. In particular, it produces no authoritative-specific or authoritative-non-specific papers in the top-10 results for any query. Hence, we do not consider this metric in our discussion henceforth. *Objd03*, which promotes papers that contain the actual keywords, performs well in terms of authoritative-specific results. The reason is that because the queries in Table II refer to fundamental areas, it happens that many important papers contain the actual keywords.

Now, let's focus on the relationship between *Obj*, *ObjInv*, and *ObjSqrtInv*, which have the common property that they only involve keyword-specific computations. In terms of the number of authoritative-non-specific papers, *Obj* and *ObjInv* are located at the two extremes. We introduced *ObjSqrtInv* as a ranking function to combine the desirable properties of both ends. As expected, *ObjSqrtInv* has a number of authoritative-non-specific papers that is between those of *Obj* and *ObjInv*. However, *ObjSqrtInv* is superior than both *Obj* and *ObjInv* in terms of average number of authoritative-specific papers, which is a highly desirable property.

The intuition behind the selection of *ObjSqrtInv* is the following. Using *ObjInv*, a too specific object may receive a high score even if it has relatively low quality and relevance. For example, a very high quality object that happens to be relevant to 10 keywords would be ranked equal to a 10 times lower-quality document that is relevant to only one keyword. Hence, taking the square root of Inverse ObjectRank serves a purpose similar to taking the logarithm of *tf* in IR to avoid assigning top score to documents that repeat many times the keywords in an adversary way. We chose square root instead of logarithm because logarithm is sensitive to the breadth of the range of the Inverse ObjectRank values. In particular, we observed that few nodes have very large Inverse ObjectRank values which have orders of magnitude difference to the top ObjectRank values. Square root is more appropriate since $\sqrt{a \cdot c}/\sqrt{b \cdot c}$ does not depend on c ($c > 0$), whereas $\log(a \cdot c)/\log(b \cdot c)$ depends on c.

On the other hand, taking the square root of ObjectRank is a bad idea, since ObjectRank is the relevance (and quality) measure, which is the primary ranking factor, and cannot be easily tricked (especially in controlled databases like bibliographical). Other ways to decrease the weight of Inverse ObjectRank were tested, like dividing (1-d) by a constant in Equation 6, but taking the square root was found to perform better.

A surprising fact is that the average number of authoritative-specific papers for *Obj* is high. The reason is that the textbook contains multiple general references for each chapter, to introduce the topic to newcomers or carry very general concepts, which would not be judged as specific by an experienced researcher. This observation is also supported by the user survey presented below.

User Survey We asked twelve users (not involved in the project), eight database professors and four database Ph.D. students in eight different universities in the US and abroad,

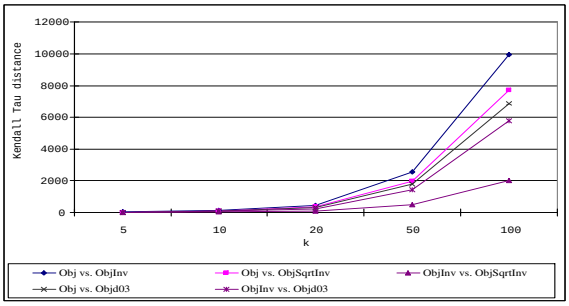

Fig. 17. Compare Results' Distances.

to rank the top-10 result lists for the five ranking functions, for various queries. The survey consisted of 9 queries, 4 of which were chapter titles of [Ramakrishnan and Gehrke 2003]. Each user/subject assigned a score between 1 and 5 to each result list for the queries/topics he/she feels comfortable with. Also, the user can specify his/her level of expertise for each topic, which is then used to weight the rating when computing average numbers. We explained to the users what is meant by authoritative-specific as opposed to authoritative-non-specific by providing the following scenario, and we asked them to evaluate according to the former.

Survey Scenario: "Let us assume you are a professor and you need to give a reading list to a first year graduate student who starts research on a topic, say "XML database storage". Being a first year student, he/she likely has no background knowledge on database issues pertaining to XML and semistructured data in general. In this case, you may want to provide an authoritative papers list where it is OK (indeed desirable) to include a few seminal papers on XML and semistructured databases, even though they may not be related to storage in particular. Such seminal papers are a good starting point for the student. These papers are authoritative-non-specific papers. Instead, our survey asks for authoritative-specific papers. Now assume that you produce a reading list for someone (perhaps yourself) who already knows the basics of XML databases and of conventional (relational) storage systems. You now care about the specific papers in XML storage, in particular."

The average ratings are shown in Table III. We observe that *ObjSqrtInv* has the highest average rating, which is consistent with our expectation that *ObjSqrtInv* outperforms other metrics because of its balance between authority and specificity. We also see that *Obj*, which lacks a specificity factor, received low ratings in contrast to Table II, where it received a high score due to the reasons mentioned above.

Surprisingly, *Objd03* received a high average rating, although setting $d = 0.3$ greatly degrades the authority flow factor and promotes results that contain the actual keywords. The reason of the high average rating is that some subjects did not have knowledge of the best papers for a topic and instead they seem to have judged by the titles of the papers and the presence of the keywords in them.

Distance Between Specificity Metrics In this experiment, we perform a quantitative comparison between the above ranking functions using the Kendall Tau distances between the generated result lists. Since the two top-k lists are not permutations of each other, we use the extended Kendall Tau definition of Fagin et al. [Fagin et al. 2003]. The average

Kendall Tau distances between the most interesting pairs of ranking functions over 100 queries are shown in Figure 17, as a function of the lists length k. Notice that as expected, there is a large distance between *Obj* and *ObjInv* but a smaller distance between *Obj* and *ObjSqrtInv*. We do not include the distance between *Obj* and *ObjOverGlobal* since their results are often disjoint hence resulting in very large distances.

8. PERFORMANCE EXPERIMENTS

In this section we experimentally evaluate the system and show that calculating the authority flows is feasible, both in the preprocessing and in the query execution stage. We present the results for ObjectRank which can be extened for Inverse ObjectRank as well. For the evaluation we use three real and a set of synthetic datasets: COMSOC is the dataset of the publications of the IEEE Communications Society [12], which consists of $55,000$ nodes and $165,000$ edges. DBLPreal and DBLPreal2 are a subset and the complete DBLP dataset respectively. DBLPreal consists of the publications in twelve database conferences. DBLPreal contains $13,700$ nodes and $101,500$ edges, whereas DBLPreal2 has $859,300$ nodes and $2,741,000$ edges. In addition, we also created a set of artificial datasets shown in Table IV, using the words of the DBLP dataset. The outgoing edges are distributed uniformly among papers, that is, each paper cites on average 10 other papers. The incoming edges are assigned by a non-uniform random function, similar to the one used in the TPC-C benchmark [13], such that the top-10% of the most cited papers receive 70% of all the citations.

name	*#nodes*	*#edges*
DBLP30	3,000	30,000
DBLP100	10,000	100,000
DBLP300	30,000	300,000
DBLP1000	100,000	1,000,000
DBLP3000	300,000	3,000,000

Table IV. Synthetic Datasets.

To store the databases in a RDBMS, we decomposed them into relations according to the relational schema shown in Figure 18. Y is an instance of a conference in a particular year. PP is a relation that describes each paper *pid2* cited by a paper *pid1*, while PA lists the authors *aid* of each paper *pid*. Notice that the two arrows from P to PP denote primary-to-foreign-key connections from *pid* to *pid1* and from *pid* to *pid2*. We ran our experiments using the Oracle 9i RDBMS on a Xeon 2.2-GHz PC with 1 GB of RAM. We implemented the preprocessing and query-processing algorithms in Java, and connect to the RDBMS through JDBC.

The experiments are divided into two classes. First, we measure how fast the ObjectRank Execution module (Figure 9) calculates the ObjectRanks for all keywords and stores them into the ObjectRank Index, using the *CreateIndex* algorithm of Figure 10. The size of the ObjectRank Index is also measured. This experiment is repeated for various values of *epsilon* and *threshold*, and various dataset sizes. Furthermore, the General ObjectRank algorithm is compared to the almost-DAG algorithm, and the effect of using various initial

[12] http://www.comsoc.org
[13] http://www.tpc.org/tpcc/

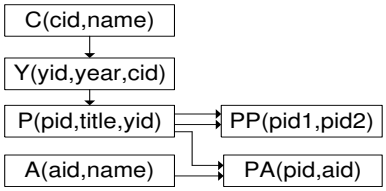

Fig. 18. Relational schema.

ObjectRank values is evaluated. Second, in Section 8.2 the Query module (Figure 9) is evaluated.

threshold	*time (sec)*	*nodes/keyword*	*size (MB)*
0.3	3702	84	2.20
0.5	3702	67	1.77
1.0	3702	46	1.26

Table V. Index Creation for DBLPreal for $epsilon = 0.1$

threshold	*time (sec)*	*nodes/keyword*	*size (MB)*
0.01	20036	70831	1854
0.03	20036	45445	1189
0.1	20036	26968	706

Table VI. Index Creation for DBLPreal2 for $epsilon = 0.05$

8.1 Preprocessing stage

General ObjectRank algorithm. Tables V, VI and VII show how the storage space for the ObjectRank index decreases as the ObjectRank *threshold* of the stored objects increases, for the real datasets. Notice that DBLPreal and COMSOC have $12,341$ and $40,577$ keywords respectively. Also notice that much fewer nodes per keyword have ObjectRank above the $threshold$ in COMSOC, since this dataset is more sparse and has more keywords. The time to create the index does not change with *threshold* since *threshold* is not used during the main execution loop of the CreateIndex algorithm. Tables VIII, IX and X show how the index build time decreases as *epsilon* increases. The reason is that fewer iterations are needed for the algorithm to converge, on the cost of lower accuracy of the calculated ObjectRanks. Notice that the storage space does not change with *epsilon*, as long as $epsilon < threshold$.

Table XI shows how the execution times and the storage requirements for the ObjectRank index scale with the database size for the DBLP synthetic datasets for $epsilon = 0.05$ and $threshold = 0.1$. Notice that the average number of nodes having ObjectRank higher than the *threshold* increases considerably with the dataset size, because the same keywords appear multiple times.

General ObjectRank vs. almost-DAG algorithm. Figure 19 compares the index creation time of the General ObjectRank algorithm (*Gen-OR*) and two versions of the almost-DAG algorithm, on the DBLP1000 dataset, for various number of backnodes. The *algebraic*

threshold	*time (sec)*	*nodes/keyword*	*size (MB)*
0.05	80829	9.4	1.17
0.07	80829	8.3	1.08
0.1	80829	7.7	1.03

Table VII. Index Creation for COMSOC for $epsilon = 0.05$

epsilon	*time (sec)*	*nodes/keyword*	*size (MB)*
0.05	3875	67	1.77
0.1	3702	67	1.77
0.3	3517	67	1.77

Table VIII. Index Creation for DBLPreal for $threshold = 0.5$

epsilon	*time (sec)*	*nodes/keyword*	*size (MB)*
0.05	20036	26968	706
0.1	18878	26968	706
0.5	16773	26968	706

Table IX. Index Creation for DBLPreal2 for $threshold = 0.1$

version (*Alg-A-DAG*) precisely solves the $\mathbf{c} = \overline{\mathbf{C}} \cdot \mathbf{c} + \overline{\mathbf{r}'}$ system using an off the self algebraic solver. The *PageRank* version (*PR-A-DAG*) solves this system using the PageRank [Brin and Page 1998] iterative method. The measured times are the average processing time for a single keyword and do not include the time to retrieve the base-set from the inverted text index, which is common to all methods. Also, the time to calculate $\mathbf{C}$ is omitted, since it $\mathbf{C}$ is calculated once for all keywords, and it requires a single pass over the graph. The *Iterative part* of the execution times corresponds to the one pass we perform on the DAG subgraph to calculate $\mathbf{r}'$ for almost-DAG algorithms, and to the multiple passes which consist the whole computation for the General ObjectRank algorithm.

Also, notice that $epsilon = 0.1$ for this experiment (the *threshold* value is irrelevant since it does not affect the processing time, but only the storage space). The time to do the topological sorting is about 20 sec which is negligible compared to the time to calculate the ObjectRanks for all keywords.

Initial ObjectRanks. This experiment shows how the convergence of the General ObjectRank algorithm is accelerated when various values are set as initial ObjectRanks. In particular, we compare the naive approach, where we assign an equal initial ObjectRank to all nodes, to the global-as-initial approach, where the global ObjectRanks are used as initial values for the keyword-specific ObjectRank calculations. We found that on DBLPreal (COMSOC), for $epsilon = 0.1$, the naive and global-as-initial approaches take 16.3 (15.8) and 12.8 (13.7) iterations respectively.

Furthermore, we evaluate the space vs. time tradeoff described in Section 6.6. Table XII shows the average number of iterations for $epsilon = 0.1$ on DBLPreal and COMSOC for various values of the precomputed list length p.

8.2 Query stage

Figure 20 shows how the average execution time changes for varying number of requested results k, for two-keyword queries on DBLPreal. The results for DBLPreal2 and COMSOC are similar. We used the index table created with $epsilon = 0.1$ (0.05) and $threshold = 0.3$. The times are averaged over 100 repetitions of the experiment. Notice that the time

epsilon	*time (sec)*	*nodes/keyword*	*size (MB)*
0.05	80829	7.7	1.03
0.07	77056	7.7	1.03
0.1	74337	7.7	1.03

Table X. Index Creation for COMSOC for $threshold = 0.1$

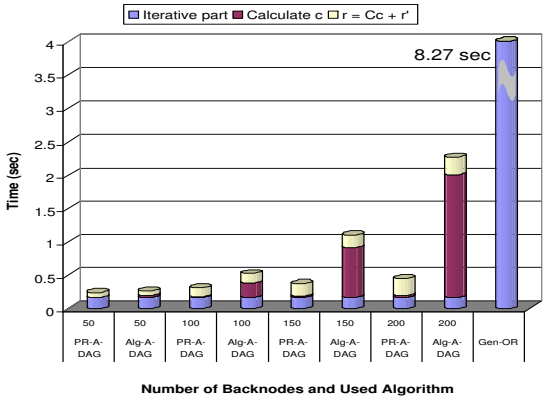

Fig. 19. Evaluate almost-DAG algorithm.

dataset	*time (sec)*	*nodes/keyword*	*size (MB)*
DBLP30	2933	6	0.3
DBLP100	11513	21	0.7
DBLP300	45764	65	1.7
DBLP1000	206034	316	7.9
DBLP3000	6398043	1763	43.6

Table XI. Index Creation for Synthetic Datasets.

does not increase considerably with k, due to the fact that about the same number of random accesses are needed for small k values, and the processing time using the Threshold Algorithm is too small. Figure 21 shows that the execution time increases almost linearly with the number of keywords, which again is due to the fact that the disk access time to the ObjectRank lists is the dominant factor, since the processing time is too small. Finally, notice that the execution times are shorter for OR semantics, because there are more results, which leads to a smaller prefix of the lists being read, in order to get the top-k results.

9. RELATED WORK

We first present how state-of-the-art works rank the results of a keyword query, using traditional IR techniques and exploiting the link structure of the data graph. Then we discuss about related work on the performance of link-based algorithms.

Traditional IR ranking. Currently, all major database vendors offer tools [Ora 2007; 2007; 2007] for keyword search in single attributes of the database. That is, they assign a score to an attribute value according to its relevance to the keyword query. The score is calculated using well known ranking functions from the IR community [Salton 1989], although their precise formula is not disclosed. Recent works [Bhalotia et al. 2002; Hris-

List length p	*iterations*
13700	1
13000	1.2
8000	1.8
2500	3
800	8.7
100	13.3
0	16.3

(a) DBLPreal

List length p	*iterations*
55000	1
54000	2.9
30000	5.3
13000	6.5
1600	7.8
400	10.7
25	13
0	15.8

(b) COMSOC

Table XII. Number of iterations for various lengths of precomputed lists

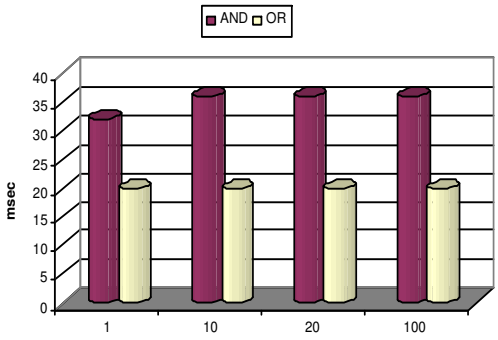

Fig. 20. Varying k in DBLPreal.

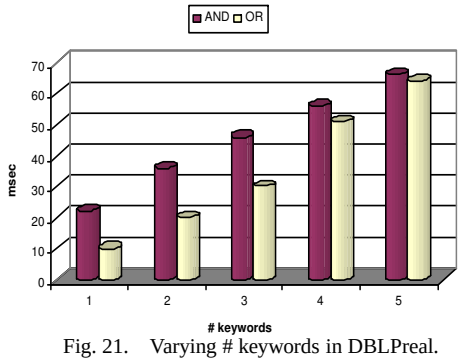

Fig. 21. Varying # keywords in DBLPreal.

tidis and Papakonstantinou 2002; Hristidis et al. 2003; Agrawal et al. 2002] on keyword search on databases, where the result is a tree of objects, either use similar IR techniques [Bhalotia et al. 2002], or use the simpler boolean semantics [Hristidis and Papakonstantinou 2002; Hristidis et al. 2003; Agrawal et al. 2002], where the score of an attribute is 1 (0) if it contains (does not contain) the keywords.

The first shortcoming of these semantics is that they miss objects that are very related to the keywords, although they do not contain them (Section 1). The second shortcoming is that the traditional IR semantics are unable to meaningfully sort the resulting objects according to their relevance to the keywords. For example, for the query ”XML”, the

paper [Gu et al. 2002] on Quality of Service that uses an XML-based language, would be ranked as high as a classic book on XML [Abiteboul et al. 2000]. Again, the relevance information is hidden in the link structure of the data graph.

The most popular specificity metric in Information Retrieval is the document length (dl). As an example, a state-of-the-art IR ranking function is [Singhal 2001]:

$$Score(a_i, Q) = \sum_{w \in Q \cap a_i} \frac{1 + ln(1 + ln(tf))}{(1-s) + s\frac{dl}{avdl}} \cdot ln\frac{N+1}{df} \tag{16}$$

where, for a word w, tf is the frequency of w in the document D, df is the number of documents in the database containing word w, dl is the size of D in characters, $avdl$ is the average document size, N is the total number of documents in the database, and s is a constant (usually 0.2). Croft [Croft 2000] and Craswell et al. [Craswell et al. 2005] present techniques on combining ranking factors.

Link-based semantics. Balmin et al. [Balmin et al. 2004] introduce the ObjectRank metric. This work extends and completes [Balmin et al. 2004] in the following ways. The specificity factor is handled and evaluated, in contrast to [Balmin et al. 2004] where the specificity factor is ignored. Inverse ObjectRank is introduced and qualitatively evaluated. Furthermore, in this work we clearly identify the ranking factors (relevance, specificity and global importance) and map them to authority flow metrics. Moreover, we explain these authority flow metrics from the perspective of information theory. We also elaborate on the combining ranking function and study techniques to weigh the query keywords. On the performance level, we present algorithms for graphs with small vertex cover and "serializable" graphs and conducted additional experiments. Finally, we have built a more complete and powerful demo available on the Web by adding adjusting parameters, and including the whole DBLP dataset and citations from Citeseer, in contrast to [Balmin et al. 2004] where a small subset of DBLP was used.

To the best of our knowledge, Savoy [Savoy 1992] was the first to use the link-structure of the Web to discover relevant pages. This idea became more popular with PageRank [Brin and Page 1998], where a global score is assigned to each Web page as we explain in Section 2. However, directly applying the PageRank approach in our problem is not suitable as we explain in Section 1. HITS [Kleinberg 1999] employs mutually dependant computation of two values for each web page: hub value and authority. In contrast to PageRank, it is able to find relevant pages that do not contain the keyword, if they are directly pointed by pages that do. However, HITS does not consider domain-specific link semantics and does not make use of schema information. The relevance between two nodes in a data graph can also be viewed as the resistance between them in the corresponding electrical network, where a resistor is added on each edge. This approach is equivalent to the random walk model [Doyle and Snell 1984].

Richardson et al. [Richardson and Domingos 2002] propose an improvement to PageRank extending the work of Bharat and Henzinger [Bharat and Henzinger 1998], where the random surfer takes into account the relevance of each page to the query when navigating from one page to the other. However, they require that every result contains the keyword, and ignore the case of multiple keywords. Haveliwala [Haveliwala 2002] proposes a topic-sensitive PageRank, where the topic-specific PageRanks for each page are precomputed and the PageRank value of the most relevant topic is used for each query. Both works apply to the Web and do not address the unique characteristics of structured databases, as

we discuss in Section 1. Furthermore, they offer no adjusting parameters to calibrate the system according to the specifics of an application.

Recently, the idea of PageRank has been applied to structured databases [Guo et al. 2003; Huang et al. 2003]. XRANK [Guo et al. 2003] proposes a way to rank XML elements using the link structure of the database. Furthermore, they introduce a notion similar to our ObjectRank transfer edge bounds, to distinguish between containment and IDREF edges. Huang et al. [Huang et al. 2003] propose a way to rank the tuples of a relational database using PageRank, where connections are determined dynamically by the query workload and not statically by the schema. However, none of these works exploits the link structure to provide keyword-specific ranking. Furthermore, they ignore the schema semantics when computing the scores. Raschid et al. [Raschid et al. 2006] and Shafer et al. [Shafer et al. 2006] have applied the PageRank ranking to rank objects of biological databases.

Geerts et al. [Geerts et al. 2004] use a set of queries to rank the values of a relational database using authority flow semantics. TrustRank [Gyongyi et al. 2004] uses the idea of Global Inverse PageRank as a heuristic for a completely different purpose than specificity. In particular, they use it to find well connected pages to use as seeds in their algorithms. Faloutsos et al. [Faloutsos et al. 2004] find the connection subgraph between two graph nodes by maximizing the electric current between the nodes, where each edge of the data graph is represented by an electric resistor. This work is extended at [Tong and Faloutsos 2006] for more than two nodes and is referred to as the center-piece subgraph problem.

Performance. A set of works [Haveliwala 1999; Chen et al. 2002; Jeh and Widom 2003; Kamvar et al. 2003] have tackled the problem of improving the performance of the original PageRank algorithm. [Haveliwala 1999; Chen et al. 2002] present algorithms to improve the calculation of a global PageRank. Jeh and Widom [Jeh and Widom 2003] present a method to efficiently calculate the PageRank values for multiple base sets, by precomputing a set of *partial vectors* which are used in runtime to calculate the PageRanks. The key idea is to precompute in a compact way the PageRank values for a set of hub pages, through which most of the random walks pass. Then using these hub PageRanks, calculate in runtime the PageRanks for any base set consisting of nodes in the hub set. However, in our case it is not possible to define a set of hub nodes, since any node of the database can be part of a base set.

10. CONCLUSIONS

We presented an adjustable framework to answer keyword queries using the authority transfer paradigm, which we believe is applicable to a significant number of domains (though obviously not meaningful for every database). We showed that our framework is efficient and semantically meaningful, with an experimental evaluation and user surveys respectively.

Furthermore we presented Inverse ObjectRank, which is a link-based and keyword-specific specificity metric. We showed how Inverse ObjectRank is combined with other ranking functions to produce the results list for a keyword query. Our methods have been qualitatively evaluated using a user survey and the bibliography sections of a database textbook. We concluded that combining ObjectRank with the square root of Inverse ObjectRank produces results of highest quality. Furthermore, we built a prototype of our methods on a bibliographic database, which we made available on the Web.

REFERENCES

2007. http://msdn.microsoft.com/library/.

2007. http://technet.oracle.com/products/text/content.html.

2007. http://www-306.ibm.com/software/data/db2/extenders/overview-text.html.

ABITEBOUL, S., SUCIU, D., AND BUNEMAN, P. 2000. Data on the Web : From Relations to Semistructured Data and Xml. *Morgan Kaufmann Series in Data Management Systems*.

AGRAWAL, S., CHAUDHURI, S., AND DAS, G. 2002. DBXplorer: A System For Keyword-Based Search Over Relational Databases. *ICDE*.

AIZAWA, A. 2000. The Feature Quantity: An Information Theoretic Perspective of Tfidf-like Measures. *SIGIR*.

BALMIN, A., HRISTIDIS, V., AND PAPAKONSTANTINOU, Y. 2004. ObjectRank: Authority-Based Keyword Search in Databases. *VLDB*.

BHALOTIA, G., NAKHEY, C., HULGERI, A., CHAKRABARTI, S., AND SUDARSHAN, S. 2002. Keyword Searching and Browsing in Databases using BANKS. *ICDE*.

BHARAT, K. AND HENZINGER, M. R. 1998. Improved algorithms for topic distillation in a hyperlinked environment. In *SIGIR*.

BRIN, S. AND PAGE, L. 1998. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *WWW Conference*.

CARMEL, D., COHEN, D., FAGIN, R., FARCHI, E., HERSCOVICI, M., MAAREK, Y. S., AND SOFFER, A. 2001. Static index pruning for information retrieval systems. In *ACM SIGIR*.

CHAKRABARTI, S., DOM, B., GIBSON, D., KLEINBERG, J., RAGHAVAN, P., AND RAJAGOPALAN, S. 1998. Automatic resource compilation by analyzing hyperlink structure and associated text. *WWW Conference*.

CHEN, Y., GAN, Q., AND SUEL, T. 2002. I/O-efficient techniques for computing PageRank. *CIKM*.

CORMEN, T., LEISERSON, C., AND RIVEST, R. 1989. Introduction to Algorithms. *MIT Press*.

CRASWELL, N., ROBERTSON, S. E., ZARAGOZA, H., AND TAYLOR, M. J. 2005. Relevance weighting for query independent evidence. In *SIGIR*.

CROFT, W. B. 2000. Combining Approaches to Information Retrieval. *Advances in Information Retrieval: Recent Research from the CIIR, Kluwer, Chapter 1*.

DAR, S., ENTIN, G., GEVA, S., AND PALMON, E. 1998. DTL's DataSpot: Database Exploration Using Plain Language. *VLDB*.

DOYLE, P. G. AND SNELL, J. L. 1984. Random Walks and Electric Networks. *Mathematical Association of America, Washington, D. C.*.

FAGIN, R., KUMAR, R., AND SIVAKUMAR, D. 2003. Comparing top k lists. In *Procs.ACM-SIAM Symposium on Discrete Algorithms (SODA)*.

FAGIN, R., LOTEM, A., AND NAOR, M. 2001. Optimal Aggregation Algorithms for Middleware. *ACM PODS*.

FALOUTSOS, C., MCCURLEY, K. S., AND TOMKINS, A. 2004. Fast discovery of connection subgraphs. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*.

GEERTS, F., MANNILA, H., AND TERZI, E. 2004. Relational link-based ranking. *VLDB*.

GOLUB, G. H. AND LOAN, C. F. 1996. Matrix Computations. *Johns Hopkins*.

GU, X., NAHRSTEDT, K., YUAN, W., WICHADAKUL, D., AND XU, D. 2002. An XML-based Quality of Service Enabling Language for the Web. *Journal of Visual Languages and Computing 13(1): 61-95*.

GUO, L., SHAO, F., BOTEV, C., AND SHANMUGASUNDARAM, J. 2003. XRANK: Ranked Keyword Search over XML Documents. *ACM SIGMOD*.

GYONGYI, Z., GARCIA-MOLINA, H., AND PEDERSEN, J. 2004. Combating Web Spam with TrustRank. *VLDB*.

HAVELIWALA, T. 1999. Efficient computation of PageRank. *Technical report, Stanford University (http://www.stanford.edu/∼taherh/papers/efficient-pr.pdf)*.

HAVELIWALA, T. 2002. Topic-Sensitive PageRank. *WWW Conference*.

HRISTIDIS, V., GRAVANO, L., AND PAPAKONSTANTINOU, Y. 2003. Efficient IR-Style Keyword Search over Relational Databases. *VLDB*.

HRISTIDIS, V. AND PAPAKONSTANTINOU, Y. 2002. DISCOVER: Keyword Search in Relational Databases. *VLDB*.

HRISTIDIS, V., PAPAKONSTANTINOU, Y., AND BALMIN, A. 2003. Keyword Proximity Search on XML Graphs. *ICDE*.

HUANG, A., XUE, Q., AND YANG, J. 2003. TupleRank and Implicit Relationship Discovery in Relational Databases. *WAIM*.

HWANG, H., HRISTIDIS, V., AND PAPAKONSTANTINOU, Y. 2006. ObjectRank: A System for Authority-based Search on Databases. *Demo at SIGMOD*.

JEH, G. AND WIDOM, J. 2003. Scaling Personalized Web Search. *WWW Conference*.

KAMVAR, S., HAVELIWALA, T., MANNING, C., AND GOLUB, G. 2003. Extrapolation Methods for Accelerating PageRank Computations. *WWW Conference*.

KLEINBERG, J. M. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM 46*.

MOTWANI, R. AND RAGHAVAN, P. 1995. Randomized Algorithms. *Cambridge University Press, United Kingdom*.

RAMAKRISHNAN, R. AND GEHRKE, J. 2003. *Database Management Systems. Third Edition*. McGraw-Hill Book Co.

RASCHID, L., WU, Y., LEE, W.-J., VIDAL, M. E., TSAPARAS, P., SRINIVASAN, P., AND SEHGAL, A. K. 2006. Ranking target objects of navigational queries. In *WIDM '06: Proceedings of the eighth ACM international workshop on Web information and data management*.

RICHARDSON, M. AND DOMINGOS, P. 2002. The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank. *Advances in Neural Information Processing Systems 14, MIT Press*.

SALTON, G. 1989. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. *Addison Wesley*.

SAVOY, J. 1992. Bayesian inference networks and spreading activation in hypertext systems. *Information Processing and Management 28*, 3, 389–406.

SHAFER, P., ISGANITIS, T., AND YONA, G. 2006. Hubs of knowledge: using the functional link structure in Biozon to mine for biologically significant entities. *BMC Bioinformatics. 2006 Feb 15;7:71*.

SINGHAL, A. 2001. Modern information retrieval: a brief overview. *IEEE Data Engineering Bulletin, Special Issue on Text and Databases 24*, 4 (Dec.).

TONG, H. AND FALOUTSOS, C. 2006. Center-piece subgraphs: problem definition and fast solutions. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*.