Explaining and Reformulating Authority Flow Queries

Ramakrishna Varadarajan
Florida International University
ramakrishna@cis.fiu.edu

Vagelis Hristidis
Florida International University
vagelis@cis.fiu.edu

Louiqa Raschid
University of Maryland
louiqa@umiacs.umd.edu

Abstract

Authority flow is an effective ranking mechanism for answering queries on a broad class of data. Systems have been developed to apply this principle on the Web (PageRank and topic sensitive PageRank), bibliographic databases (ObjectRank), and biological databases (Hubs of Knowledge project). However, these systems have the following drawbacks: (a) There is no way to explain to the user why a particular result received its current score; (b) The authority flow rates, which have been shown to dramatically affect the results' quality in ObjectRank, have to be set manually by a domain expert; (c) There is no query reformulation methodology to refine the query results according to the user's preferences. In this work, we address these shortcomings by introducing a framework and algorithms to explain query results and reformulate authority flow queries based on the user's feedback. The query reformulation process can be used to learn the user's preferences and automatically adjust the authority flow rates to facilitate personalized authority flow searching. We experimentally evaluate our algorithms in terms of performance and quality.

1. Introduction

Authority flow is an effective ranking mechanism for answering queries on a broad class of data. In the context of the Web, PageRank [BP98] is used to compute a global ranking of the pages based on the hyperlink structure. ObjectRank [BHP04] applies the idea of authority flow on a data graph, where nodes represent entities like tuples, and edges represent associations like primary-to-foreign keys. In contrast to PageRank, ObjectRank provides query-specific ranking by using the query-specific nodes as the authority source (called *base set*). Another key feature of ObjectRank, as explained below, is that different edge types carry different amounts of authority. Another appropriate domain for authority flow ranking is that of biological data. The Hubs of Knowledge project [SIY06] applies the PageRank algorithm on a query-dependent subgraph of the original biological graph. Raschid et al. [RWL+06] apply PageRank and ObjectRank to answer navigational queries on biological data. We note that not all databases are appropriate for authority flow query answering. Our work is applicable to the ones that do have the notion of authority flow along their associated edges as the ones described above. Web search is out of the scope of this work, since we focus on typed domain-specific data graphs.

ObjectRank: We build our work on ObjectRank, since it is the most general of the available authority flow ranking approaches. We use a modification of ObjectRank, which we refer to as ObjectRank2, where the nodes in the base set are weighted according to Information Retrieval (IR) techniques. Consider the example of Figure 1, which illustrates a small subset of the DBLP database in the form of a labeled graph (some author, conference and year nodes are omitted to simplify the figure). Schema graphs, such as the one of Figure 2, 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. Given the subgraph of Figure 1, the "Data Cube" paper is ranked on the top, even though it does not contain the keyword "OLAP". This is a key advantage of ObjectRank compared to traditional IR approaches.

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 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. These factors are depicted in an authority transfer schema graph. Figure 3 illustrates the authority transfer schema graph used by the ObjectRank project [BHP04]. Assuming the probability that the surfer moves back to an "OLAP" node is 15% (damping factor–random jump probability [BP98]), the collective probability to move to a referenced paper is up to 85% · 70% (70% is the authority transfer rate of the citation edge as we explain below), 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 to a limit $r(v)$. Intuitively, this limit is the ObjectRank of the node.

Limitations of ObjectRank: Ranking the objects of a data graph using ObjectRank has the following limitations:

(a) There is no way to *explain* to the user why a particular result received its current score; e.g., the user may want to

see proof to believe that the "Data Cube" paper is important for "OLAP". This is even more critical in complex biological databases where objects (e.g., proteins) with no obvious connection to the query (e.g., gene "TNF") are returned. Figure 4 shows a subgraph of the biological schema graph we are using in our experiments.

(b) The authority flow rates, which have been shown to dramatically affect the results' quality of ObjectRank [BHP04,HVP06], have to be set manually by a domain expert. For instance, what is the ratio of authority flowing from a gene to a PubMed publication over that flowing to a protein at Entrez Protein?

(c) There is no *query reformulation* methodology to refine the query results according to the user's preferences. Query reformulation based on user relevance feedback is a mature and well-studied process in traditional document IR [SB90, RL03, Har88]. However, there is no work on handling user feedback for authority flow ranking systems. As we discuss below, our reformulation strategy exploits the user feedback in terms of content (in the spirit of traditional query expansion [BSA+95, MSB98, SVR83, Efth93]) as well as link structure.

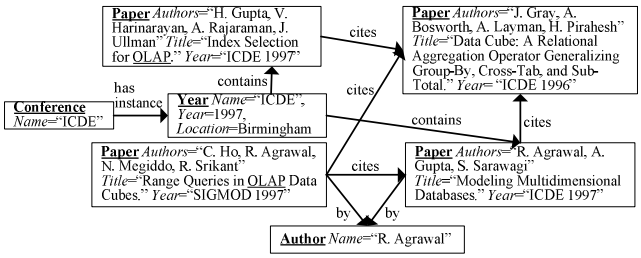


Figure 1: A subset of the DBLP graph.

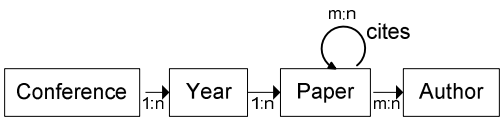


Figure 2: The DBLP schema graph.

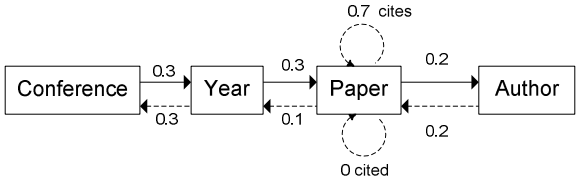


Figure 3: The DBLP authority transfer schema graph.

Contributions: This paper has the following contributions:

1. We present a methodology to explain a result of an authority flow query. For that we generate and display an *explaining subgraph*.
2. A process is presented to automatically reformulate a query based on a selection of good results by the user. We reformulate the query based both on the content, in the spirit of traditional IR query expansion, as well as the link-structure, by adjusting the authority flow rates. The explaining subgraphs of the selected results are used to capture the user's preference and build the reformulated query.

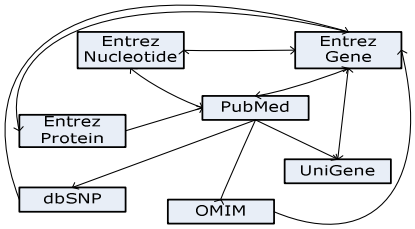


Figure 4: Subgraph of biological schema graph.

3. Efficient algorithms are presented and evaluated to explain a result, and create and evaluate the reformulated query.
4. An ObjectRank2 query and reformulation system has been built and deployed on bibliographic and biological datasets, available on the Web at *http://dbir.cis.fiu.edu/ObjectRankReformulation/*.
5. User surveys were conducted, which prove the utility of the system in explaining and reformulating queries, as well as in automatically training the authority flow rates, which was done manually before [BHP04]. .

The rest of the paper is organized as follows. Section 2 presents the framework. Section 3 defines a query and presents ObjectRank2. Section 4 presents the result explanation technique while Section 5 presents the query reformulation techniques. Section 6 present the quality and performance experiments. Section 7 describes the related work. Finally Section 8 discusses our conclusions.

2. Framework

In this section we present the framework and essential definitions, which are later used to describe our result explanation and query reformulation techniques. Note that we follow the terminology of [BHP04].

We view a database as a labeled graph, which is a model that 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.

Each node v has a role $\lambda(v)$. For instance, the ICDE conference node in Figure 1 has role "conference". 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 "$u \rightarrow v$".

The data graph can represent relational [ACD02, HP02] and XML [HPB03, GSB+03] databases, as well as the Web [BP98], although we repeat that the Web is out of the scope of this work. The *schema graph* $G(V_G,E_G)$ (Figures 2, 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. For instance, the "Entrez Gene" to "PubMed" edge in Figure 4 has role "genePubMedAssociates". 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 assignment 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. In particular, for each edge $e_G = (u \rightarrow v)$ of E_G, two *authority transfer edges*, $e_G^f = (u \rightarrow v)$ and $e_G^b = (v \rightarrow u)$ 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 $\alpha(e_G^b)$ respectively. 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.) In Balmin at el. [BHP04] the authority transfer rates for each edge type was assigned manually by a domain expert on a trial and error basis. In contrast, our techniques allow this task to be done automatically based on the user's feedback as we explain in Section 5.

Figure 3 shows the authority transfer schema graph that corresponds to the schema graph of Figure 2 (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 important papers is clearly important but citing important papers does not make a paper important.

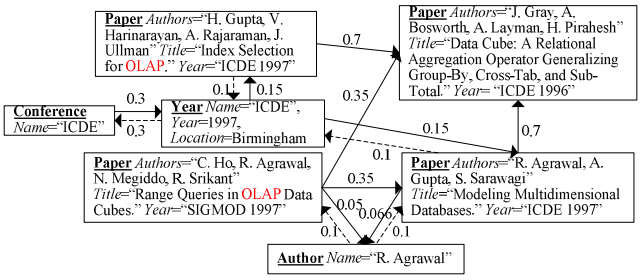


Figure 5: The DBLP Authority transfer data graph.

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

$$\alpha(e^f) = \begin{cases} \dfrac{\alpha(e_G^f)}{OutDeg\ (u, e_G^f)}, if OutDeg\ (u, e_G^f) > 0 \\ 0, if OutDeg\ (u, e_G^f) = 0 \end{cases} \quad (1)$$

where $OutDeg(u, e_G^f)$ is the number of outgoing edges from u, of type e_G^f. The authority transfer rate $a(e^b)$ is defined similarly. Figure 5 illustrates the authority transfer data graph that corresponds to the data graph of Figure 1 and the authority transfer schema graph of Figure 3. Each edge is annotated with its authority transfer rate. Notice that the sum of authority transfer rates of the outgoing edges of a node u of type $\mu(u)$ in the authority transfer data graph 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. Query Definition and ObjectRank2

In this section we define a query and describe a slightly modified version of ObjectRank originally presented in [BHP04], called ObjectRank2. The modification to the original definition is that the nodes of the base set are

weighted. The weights are computed using IR techniques for the original query and using query expansion techniques for subsequent queries as we describe in Section 5.

Keyword Query. A keyword query Q is defined as a tuple[1] of keywords $Q=[t_1,\ldots,t_m]$.

To incorporate weighing in the base set, we define the query vector as follows:

Query Vector. For each query $Q=[t_1,\ldots,t_m]$ we define a query vector $\boldsymbol{Q}=[w_1,\ldots,w_m]$ where w_i is the weight of the query keyword t_i. The initial query vector for a query is $[1,\ldots,1]$, since we assume that the query term weights are all 1. These weights change during the query expansion stage described in Section 5. The answer to $\boldsymbol{Q}$ is a list of objects with descending ObjectRank2 scores with respect to $\boldsymbol{Q}$.

ObjectRank2 is computed as follows. 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 node (database object) v_i of the base set of V and at each step, he/she follows an edge with probability d or gets bored and jumps to a node in the base set with probability $1 - d$. The ObjectRank2 value of v_i is the probability that at a given point in time, the surfer is at v_i. Given a query Q, we first find the query base set $S(Q)$, (from now on referred to simply as base set when the keyword is implied) which is the set of nodes/objects that contain at least one keyword in Q. In contrast to the original ObjectRank [BHP04], the random surfer jumps to different nodes of the base set with different probabilities. This probability for a node v is proportional to the IR score $IRScore(v,\boldsymbol{Q})$ of the node (a node is also viewed as a document–we overload symbol v in this case) given the query vector $\boldsymbol{Q}$.

$$IRScore(v,\boldsymbol{Q}) = \boldsymbol{v}\cdot\boldsymbol{Q} \qquad (2)$$

where "$\cdot$" denotes the dot product operator, $\boldsymbol{v}=[W(v,t_1),\ldots,W(v,t_m)]$ is the document vector for v, and $W(v,t)$ is the IR weight of term t for document v. $W(v,t)$ is defined using a traditional IR weighing formulas like BM25 [RW94] or Okapi [Sin01]. The latter is defined below:

$$\sum_{t\in Q,v} \ln\frac{n-df+0.5}{df+0.5}\cdot\frac{(k_1+1)tf}{(k_1(1-b)+b\frac{dl}{avdl})+tf}\cdot\frac{(k_3+1)qtf}{k_3+qtf} \qquad (3)$$

where, for a term t, tf is the frequency of t in the document v, df is the number of documents in the database containing term t, dl is the size of v in characters, $avdl$ is the average document size, n is the total number of documents in the database, and k1 (between 1.0–2.0), b (usually 0.75), and k3 (between 0–1000) are constants.

[1] We use a tuple and not a set as in [BHP04], because the order of the keywords becomes important when we introduce the notion of the weighted base set.

We normalize the IR scores of the nodes in the base set to sum to one, since they represent probabilities. The ObjectRank2 scores vector $\boldsymbol{r}^Q = [r^Q(v_1),\ldots,r^Q(v_n)]^T$ given query vector $\boldsymbol{Q}$, where $n=|V_D|$, is defined as follows:

$$\mathrm{r}^Q = dA\mathrm{r}^Q + \frac{(1-d)}{|S(Q)|}s \qquad (4)$$

where A is a $n \times n$ matrix with $A_{ij} = \alpha(e)$ if there is an edge $e(\mathrm{v_j} \rightarrow \mathrm{v_i})$ in E_D^A and 0 otherwise, d is the damping factor which controls the base set importance, and $s = [s_1, \ldots s_i \ldots, s_n]^T$ is the base set vector, where $s_i = IRScore(v_i,\boldsymbol{Q})$ if $v_i \in S(Q)$ and $s_i = 0$ otherwise. Note that the only difference to ObjectRank is the definition of the s_i's which were 0 or 1 in [BHP04].

4. Explaining Query Results

In this section we tackle the problem of explaining a query result. For instance, as discussed in Section 1, the "Data Cube" paper in Figure 1 (see Figure 5 for corresponding authority transfer data graph) is ranked high for the query "OLAP". What is the best way to explain to the user why this paper, referred to as the *target object*, received a high rank? This problem is even more critical in complex biological databases as the one of Figure 4.

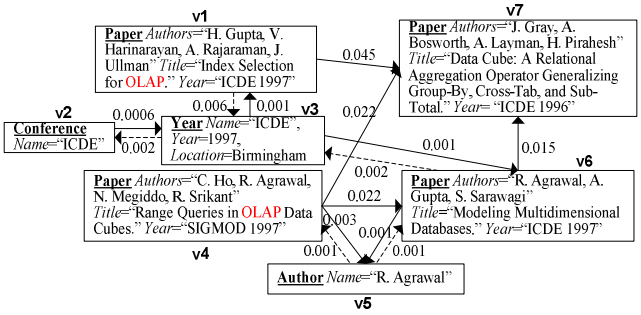


Figure 6: The DBLP Authority transfer data graph annotated with authority flows for query "OLAP".

Intuitively, we want to show to the user the paths in the authority transfer data graph D^A that authority traversed to reach the target object v, starting from the nodes in the base set $S(Q)$. For that, we create an *explaining subgraph* G_v^Q of D^A that contains all edges that transfer authority to v given Q, and every edge in G_v^Q is annotated with the amount of authority that flows on this edge and eventually reaches v.

We create G_v^Q in two stages:

(i) *Construction stage:* G_v^Q contains all nodes and edges of D^A that are part of a directed path going from the base set $S(Q)$ to v. That is, G_v^Q contains all edges that can potentially carry authority flow to v.

(ii) *Flow adjustment stage:* We compute the *explaining authority flows* on the edges of G_v^Q. The explaining authority flow *Flow(e)* of an edge *e* is the amount of authority flow that is transferred through *e* and eventually reaches *v*, on D^A for *Q*.

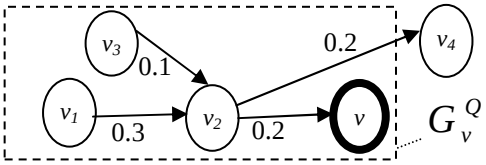


Figure 7: Intuition behind flow adjustment.

The construction stage is straightforward and is achieved as follows: We first construct the temporary subgraph D_v, starting from the target node *v* and traversing edges of D^A following the edges in the opposite direction in a breadth first manner (depth first would also work) until no more edges can be traversed. Then, we start from the authority sources (base set nodes) of D_v and traverse the edges of D_v in the forward direction until no more edges can be traversed. All nodes and edges traversed in the forward stage are added to the explaining sub graph G_v^Q.

The flow adjustment stage is more challenging because we have to adjust the "original" edge authority flows for *Q* to subtract the authority flow not reaching to *v*. For instance, in Figure 7 we must subtract from the edge flows the amount that will eventually "leak" out of G_v^Q through $v_2 \rightarrow v_4$. By "original" flows we refer to the authority flows at convergence state in D^A for ObjectRank2 execution for query *Q*. The original flow for edge $v_i \rightarrow v_j$ is:

$$Flow_0(v_i \rightarrow v_j) = d \cdot \alpha(v_i \rightarrow v_j) \cdot r^Q(v_i) \qquad (5)$$

where $a(v_i \rightarrow v_j)$ is the authority transfer rate of edge $e = (v_i \rightarrow v_j)$ in D^A according to Equation 1.

Figure 6 illustrates the original authority flows for $d = 0.85$ and query Q=["OLAP"], on the authority transfer data graph of Figure 5. The computed ObjectRank2 scores vector $\mathbf{r}^Q = [0.076, 0.002, 0.009, 0.076, 0.017, 0.025, 0.083]^T$, after 5 iterations.

It is more intuitive to view the problem as adjusting the edge flows instead of adjusting the node scores, although the adjusted node scores can be easily computed given the edge flows in the end.

One could think of simply reducing the flow on an incoming edge $v_i \rightarrow v_j$ of G_v^Q proportionally to the ratio of the outgoing flow of v_j going outside G_v^Q. However, this approach will fail if there are cycles in G_v^Q, since adjusting the flow of an edge can have a ripple effect. Hence, an iterative method is used, in the spirit of the PageRank/ObjectRank computation. In particular, for every node *u*, with the exception of the target node *v*, we iteratively reduce its incoming flows proportionally to the flow going from *u* towards nodes outside of G_v^Q. We do not adjust the incoming flows of the target node *v*, as the purpose of the explaining subgraph is to explain to the user the total authority that *v* receives from other nodes in D^A. We assume all edges are bidirectional (arbitrarily small flow rates can be assigned to direction of small importance) to guarantee convergence as proved by Theorem 1.

For instance, for the explaining subgraph in Figure 7 with target node *v*, where we assume *d=1* (i.e., nodes pass all their authority to their neighbors) and all edges are of the same type, we adjust the original edge flows of $v_1 \rightarrow v_2$ and $v_3 \rightarrow v_2$ as follows: Half of the flow going through these edges goes through $v_2 \rightarrow v$ and half through $v_2 \rightarrow v_4$. Since $v_2 \rightarrow v_4$ is outside G_v^Q, we cut the flows of $v_1 \rightarrow v_2$ and $v_3 \rightarrow v_2$ to half, i.e., to 0.15 and 0.05 respectively. This process is repeated iteratively for all edges in G_v^Q until the computation converges. Note that the flow on edges $v_i \rightarrow v$, i.e., edges that end at *v*, are not adjusted.

Details of adjustment stage: The details of the adjusting algorithm are as follows: For each node v_k in G_v^Q, let $O(v_k)$ be the summation of all outgoing flows of v_k in G_v^Q and $I(v_k)$ be the summation of all incoming flows of v_k in G_v^Q (we consider all incoming edges in G_v^Q and not D^A since Observation 1 below shows that both are equal). It is

$$I(v_k) = \sum_{(v_j, v_k) \in G_v^Q} Flow(v_j \rightarrow v_k) \qquad (6a)$$

$$O(v_k) = \sum_{(v_k, v_j) \in G_v^Q} Flow(v_k \rightarrow v_j) \qquad (6b)$$

Observation 1: *There is no incoming edge $v_i \rightarrow v_j$ with non-zero authority flow, where v_j is in G_v^Q but v_i is outside G_v^Q. If such en edge existed, it would have been included to G_v^Q during the construction stage.* □

As mentioned before, our goal is to compute the factor $h(v_k)$ by which the incoming flow $I(v_k)$ of each node v_k must be reduced to be consistent with the reduced outgoing flow $O(v_k)$ of v_k in G_v^Q. It is:

$$Flow(v_j, v_k) = h(v_k) \cdot Flow_0(v_j, v_k) \qquad (7)$$

Intuitively, this factor $h(v_k)$ is computed by the ratio of $r^Q{}'(v_k)$ and $r^Q(v_k)$ which are the ObjectRank score of v_k in G_v^Q (the "original" score) and D^A respectively. Hence, for a node v_k:

$$r^Q{}'(v_k) = \frac{O(v_k)}{d} \qquad (8)$$

Explain-ObjectRank(Target Object v, Graph D^A, Base Set $S(Q)=\{s_1,\ldots,s_n\}$, Threshold T) {

/*Construction Stage */
1)Create a temporary subgraph D_v by executing breadth-first search on D^A with v as the root node, traversing edges in opposite direction;
2)Create explaining subgraph, G_v^Q by executing breadth-first search on D_v with the nodes in base set S(Q) as root nodes, traversing edges in right direction;
/*Flow Adjustment Stage */
3)For each edge v_i->v_j in G_v^Q, compute $Flow_0(v_i \to v_j)$ using Equation 5;
4)For each node v_k in G_v^Q set $h(v_k)=1$;
5)While not converged do
    For each node v_k in G_v^Q except v do
        Compute $h(v_k)$ using Equation 10;
6)Update the Flow of each edge in G_v^Q using Equation 7;
7)Return G_v^Q ;

Figure 8: Algorithm to Compute Flows in Explaining Subgraph.

$$h(v_k)=\frac{r^{Q'}(v_k)}{r^{Q}(v_k)} \qquad (9)$$

Combining Equations 6b, 7, 8 and 9, we get the following fixpoint equation for the computation of $h(v_k)$.

$$h(v_k)=\frac{\sum_{(v_k,v_j)\in G_v^Q}\left(h(v_j)\cdot Flow_0(v_k,v_j)\right)}{d\cdot r^Q(v_k)}$$

We rewrite this equation using Equation 5:

$$h(v_k)=\frac{\sum_{(v_k,v_j)\in G_v^Q}\left(h(v_j)\cdot d\cdot \alpha(v_k\to v_j).r^Q(v_k)\right)}{d*r^Q(v_k)}$$

which then becomes

$$h(v_k)=\frac{d\cdot r^Q(v_k)\cdot \sum_{(v_k,v_j)\in G_v^Q}\left(h(v_j)\cdot \alpha(v_k\to v_j)\right)}{d\cdot r^Q(v_k)}$$

and finally,

$$h(v_k)=\sum_{(v_k,v_j)\in G_v^Q}\left(h(v_j)\cdot \alpha(v_k\to v_j)\right) \qquad (10)$$

Observation 2: *The "original" ObjectRank2 scores are not used in computing the reduction factor $h(v_k)$.* □

Implementation note: While executing the iterative algorithm defined by Equation 10, we need to store the current $h(v_k)$ factor for each node v_k. The current values of the incoming flows do not need to be stored since they can be on-the-fly computed from the original values using Equation 7.

Theorem 1: *Iteratively computing Equation 10 on the explaining subgraph converges.*

Proof: The fixpoint computation of Equation 10 is equivalent to the PageRank computation, if we replace incoming by outgoing edges and remove the damping factor. The PageRank computation has been shown to converge if the graph is aperiodic and irreducible [MR95]. The former is generally satisfied, whereas the latter is satisfied for connected graphs. The explaining subgraph is connected due to its construction method – all nodes are connected to the target node. To guarantee convergence, we always consider a non-zero reverse direction edge type for every edge type. Furthermore, there are no flow sinks [BP98] since there is a path from every node to the target node. □

When it converges, we update all edge flows using Equation 7. Figure 8 shows the steps of the algorithm to create the explaining subgraph.

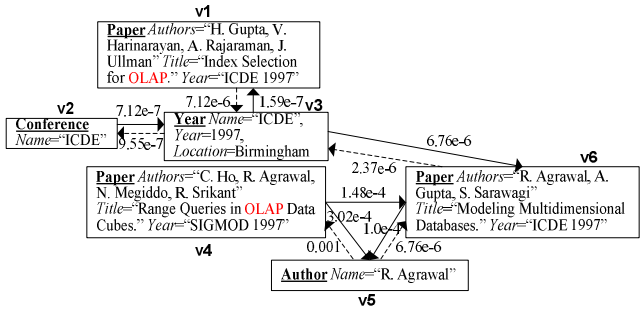


Figure 9: Explaining Subgraph for "Range Queries in OLAP" paper in Figure 6.

Example 1. *Figure 9 shows the explaining subgraph for Q=["OLAP"] and target object v4 after 5 iterations of Equation 10. Note that the "Data Cube" paper (see Figure 6) is not in G_v^Q, since there is no path from that paper to v4.*

Notice that the incoming flows of the target object v4 are the same as the original ones of Figure 6. The computed reduction factors after 5 iterations are as follows: h(v1)=1.59e-4, h(v2)=4.77e-4, h(v3)=0.0011, h(v4)=1.0, h(v5)=0.1006 and h(v6)=0.0067. Note that h(v4) is 1 as v4 is the target object which implies that its incoming flow from v5 is not adjusted as shown in Figure 9. Also notice that the incoming flow reduction factor of v1 is small because its incoming flow needs to be reduced proportionally to eliminate the large flow from v1 to v7 (see Figure 6), which is going outside of G_v^Q. This in turn creates a ripple effect reducing the incoming flows of nodes v3 and v2, according to their reduction factors. □

The explaining subgraph G_v^Q can be very large which would make its generation slow and its display to the user, impossible. Hence, in practice we limit the radius of G_v^Q to L (longer paths are generally unintuitive [CQ69] and carry

less authority) and only keep the paths with high authority flow. We apply these techniques in our online demo. We have found that a relatively small L (e.g., $L=3$) value is adequate to effectively explain a result and produce useful reformulations (see Section 5).

5. Query Reformulation

Query reformulation using relevance feedback has been well studied in traditional IR [SB90, RL03, Efth93, BSA+95, Har88], where query expansion has been the dominant strategy. That is, keywords are added to the original query according to the user's feedback. Such techniques are not adequate for ObjectRank2, since they ignore the link-structure of the graph which plays a key role in the ranking.

In this section we tackle for the first time the problem of reformulating authority flow queries. We propose an automatic query reformulation technique given the results selected by the user. For instance, if the user selects the "Range Queries in OLAP" paper in Figure 5 as a relevant object, what is the best way to reformulate the query using this paper (referred as *feedback object*)? The explaining subgraph described in Section 4 is a key structure for query reformulation since *a "vote" of the user for feedback object v can be viewed as "vote" of the user for the explaining subgraph* G_v^Q *of v*.

Overview of process: First, the system computes the top-*k* objects with the highest ObjectRank2 values. The user marks a result object *v* (or multiple objects) as relevant – user's click-through could be used to implicitly derive such markings. Then the explaining subgraph G_v^Q of *v* is computed. Based on the content and link-structure of G_v^Q we reformulate the initial query. In particular, the *Content-based* component (Section 5.1) of the reformulation is inspired by traditional query expansion ideas and leads to a query expansion; whereas the *Structure-based* component (Section 5.2) adjusts the authority transfer rates of the authority transfer schema graph based on the edge types in G_v^Q. The two reformulation components can be combined.

5.1 Content-based Reformulation

According to traditional reformulation techniques, the terms in the feedback object *v* (viewed as a document) should be added, appropriately weighted, to the original query. However, due to the nature of authority flow ranking, we extend this idea to also include terms in the objects that transfer high authority to *v*. These objects are the nodes of the explaining graph G_v^Q. The weight of an expansion term *t* is proportional to the flow that the nodes that contain *t* pass to *v*, that is, the outgoing flow of these nodes in G_v^Q. We consider a single feedback object *v* – we extend to multiple feedback objects in Section 5.3.

A term *t* is weighted according to its distance from *v* and the amount of authority it transfers to *v*, as shown in Equation 11. The authority flow a node transfers to *v* is its outgoing flow in the explaining graph G_v^Q as explained in Section 4.

$$w^f(t) = \sum_{v_k \in G_v^Q \wedge t \in v_k} \left((C_d)^{D(v_k,v)} \cdot \sum_{(v_k,v_j) \in G_v^Q} Flow(v_k \rightarrow v_j) \right) \quad (11)$$

where $0 \leq C_d \leq 1$ is the *decay factor* (in the spirit of XRANK [GSB+03]) which is typically set to 0.5, and $D(v_k,v)$ is the distance (length in number of edges) of v_k from *v*. Note that if v_k is *v*, then we use $d \cdot \sum_{(vj,vk) \in G_v^Q} Flow(v_j \rightarrow v_k)$ instead of $\sum_{(v_k,v_j) \in G_v^Q} Flow(v_k \rightarrow v_j)$, since the outgoing flow of *v* is not specified in G_v^Q.

We select the top-*s* terms *Z* with highest weight (ignoring stop words) and add them, after normalizing them as explained below, to the original query vector $\mathbf{Q}_0$. The reformulated query vector $\boldsymbol{Q}_i$ at iteration *i* is defined as

$$\mathrm{Q}_i = \begin{cases} \mathrm{Q}_{i-1} + C_e \cdot \sum_{t \in Z} w^f(t) \cdot \mathrm{t}, & i > 1 \\ \mathrm{Q}_0, & i = 0 \end{cases} \quad (12)$$

where **t** is the vector of term *t* (as in the vector space model [Sin01]), and $0 \leq C_e \leq 1$ is the *expansion factor*, typically 0.5, used to scale the weights of new terms (as well as new weights of old terms) with respect to the terms present in current query vector.

Normalization of term weights: The computed term weights using Equation 11 must be normalized with respect to the weights in the current query vector. This done as follows:

1) Compute the average a_v of the term weights of the current query vector $\boldsymbol{Q}_i$,

2) Let $x = \max_{t \in S} w^f(t)$ be the maximum weight of any term in *Z*. Let t_s be the term that has this weight.

3) We normalize the term weight of terms in *Z* such that $w^f(t_s)$ becomes a_v. That is, we multiply all term weights in *Z* with $\frac{a_v}{x}$.

Example 2. *Consider the authority transfer data graph of Figure 5, query Q=["OLAP"], and feedback object, v is the "Range Queries in OLAP" paper. The explaining subgraph* G_v^Q *(Figure 9) is created. Using Equation 10, and assuming* C_d *and* C_e *are 0.5, the top-5 new terms are olap(1.0), cubes(0.99), range(0.99), multidimensional(0.05) and modeling(0.05). Note that the terms in the feedback object (target object of* G_v^Q*) generally get a higher weight due to the decay factor* C_d*. The reformulated query vector* $\boldsymbol{Q}$ *computed by Equation 11 is [olap, cubes, range,*

multidimensional, modeling] = [2.0, 0.99, 0.99, 0.05, 0.05].□

5.2 Structure-based Reformulation

The structure-based reformulation adjusts the authority transfer rates based on the explaining subgraph G_v^Q. We consider a single feedback object v – we later extend to multiple feedback objects in Section 5.3. Intuitively, if edges of an edge type e_G carry large authority in G_v^Q then the user probably believes e_G is an important edge type for the query. We boost the authority transfer rate of each edge type present in G_v^Q according to the authority it transfers (to the feedback object v). The reformulated authority transfer rate $a'(e_G)$ of edge type e_G is computed by

$$\alpha'(e_G) = \left(1 + C_f \cdot \sum_{(v_k,v_j)\in G_v^Q \wedge (v_k,v_j) has\ \ type\ \ e_G} Flow(v_k \to v_j)\right) \cdot \alpha(e_G) \qquad (13)$$

where $0 \le C_f \le 1$ is the *authority transfer rate adjustment factor*, typically set to 0.5, used to scale the authority transfer rates with respect to their previous values, $\alpha(e_G)$ is the previous authority flow rate of edge type e_G. The factor $F(e_G) = \sum_{(v_k,v_j)\in G_v^Q \wedge (v_k,v_j) has\ \ type\ \ e_G} Flow(v_k \to v_j)$ in Equation 13 is normalized before used, as explained below.

Normalization: In order to have a controlled modification of authority transfer rates we do the normalization as follows:

1) For each edge type e_G we normalize the factor $F(e_G)$ by setting its maximum value M for an edge type to 1. Then divide others by M.

2) Compute $a'(e_G)$ for each edge type e_G using Equation 13 and the normalized values from Step 1.

3) Normalize $a'(e_G)$ as in Step 1, such that $0 \le \alpha'(e_G) \le 1$.

4) Finally we once again adjust $a'(e_G)$ of each edge type such that the sum of authority transfer rates of the outgoing edges of every schema node in the authority transfer schema graph is less than or equal to 1, which is required for the convergence of ObjectRank2.

Example 2 (cont'd). *The authority transfer rates of the original query are [PP,PP',PA,AP,CY,YC,YP,PY] = [0.7,0.0,0.2,0.2,0.3,0.3,0.3,0.1]. Using Equation 13 and the normalization process, the reformulated authority transfer rates are [0.67,0.0,0.24,0.16,0.24,0.24,0.24,0.08]. Notice that the transfer rates of PA and AP edge types are increased and decreased respectively as they carry greater and lesser authority to the feedback object respectively.* □

5.3 Multiple Feedback Objects

When the user selects multiple feedback objects U, we combine the expansion term weights (in Content-based reformulation) and the flow weights (in Structure-based reformulation) across the explaining subgraphs of different feedback objects using a monotone aggregation function. Typical choices are sum, min, max and average. We use summation in our user surveys and experiments. That is, for content-based reformulation we add the expansion term weights.

$$w^f(t) = \sum_{v_i \in U} w_i^f(t) \qquad (14)$$

where $w_i^f(t)$ is the weight for term t for feedback object v_i, computed by Equation 11.

For Structure-based reformulation, similarly we add the $F(e_G)$ factors for each edge types.

$$F(e_G) = \sum_{v_i \in U} F_i(e_G) \qquad (15)$$

where $F_i(e_G)$ is the sum of flows for edges of type e_G for feedback object v_i, computed by Equation 13.

We follow the same normalization process after aggregating the weights to yield the necessary query reformulation.

6. Experiments

We experimentally evaluate our algorithms in terms of quality and performance. We conducted user surveys to evaluate the quality of our query reformulation techniques. We also evaluate the performance of our algorithms and show that explaining query results and reformulating authority flow queries are feasible over large graphs. This section is organized as follows: First we briefly describe the datasets used for evaluation and then Sections 7.1 and 7.2 present the user surveys and the performance experiments respectively.

Datasets: We use four real datasets (Table 1). DBLPcomplete and DBLPtop are the complete DBLP dataset and a databases-related subset respectively. We shredded the downloaded DBLP file into the relational schema of Figure 2. DS7, whose schema is shown in Figure 4, is a collection of biological sources downloaded from PubMed. DS7cancer is a subset of DS7 consisting of PubMed publications related to "cancer" and all biological entities related to these publications.

Name	*#nodes*	*#edges*	*Size* (*MB*)
DBLPcomplete	876,110	416,662,6	3950
DBLPtop	22,653	166,960	136
DS7	699,199	353,375,6	2189
DS7cancer	37,796	138,146	111

Table 1: Real and Synthetic Datasets.

6.1 User Surveys

We used DBLPtop for our user surveys and not DBLPcomplete since on-the-fly ObjectRank2 executions on the latter are slow and survey subjects would be irritated.

The first phase (Section 6.1.1) was conducted at Florida International University (FIU) involving five professors and PhD students from the database lab, who were not involved with the project. The goal of this survey was to compare content-based, structure-based, and content & structure-based reformulations. The result was that structure-based reformulation is superior. The second phase (Section 6.1.2) focused on structure-based reformulation and involved 10 FIU and outside (including IBM TJ Watson and Almaden) database researchers, not involved in the project. In both phases we also measure the capability of our system to discover the authority transfer rates set by a domain expert.

6.1.1. Internal Survey.

First we measure the precision/recall of our relevance feedback algorithms using the standard relevance feedback evaluation method of *residual collection* [RL03, SB90]. Then we show how our structure-based reformulation component can be used to automatically train the authority transfer rates of the DBLP authority transfer schema graph (Figure 3). We measure the quality of this training assuming the ground truth rates for this dataset are the ones found at [BHP04].

The residual collection method [RL03, SB90] can be summarized as follows: All objects seen by the user or marked as relevant are removed from the collection and both the initial and all reformulated queries are evaluated using the residual collection. We use the average precision as the evaluation measure. Note that the recall is the same as the precision in our case since we limit the output results to k. We report the survey results for 4 relevance feedback iterations and for the following 3 settings: i) Content-Only reformulation (C_f=0&C_e=0.2), ii) Content & Structure-based reformulation (C_f =0.5& C_e =0.2) and iii) Structure-Only reformulation (C_f =0.5& C_e =0). (We have found that these values of C_f and C_e are appropriate for this dataset.) The decay factor C_d is set to 0.5.We use $L=3$ to limit the size of the explaining subgraph as explained in Section 4, We initialize the authority transfer rates of each edge type to 0.3. Figure 10 shows the survey results. We see that the structure-only reformulation performs the best. Content-based reformulation is not effective in our setting because the users are domain experts and hence know the right keywords, i.e., traditional query expansion is not effective. Note that in a different domain the results could vary.

Next we evaluate the effectiveness of structure-based reformulation to automatically train the authority transfer rates of the DBLP authority transfer schema graph and compare the learned weights to the ones of [BHP04], which we view as ground truth. The rates there were assigned manually by domain experts in a trial and error manner. We start by setting the transfer rates of all edge types to 0.3. We again limit the length of paths of the explaining graph with $L=3$. Let UserVector[PP,PP′,PA,AP,CY,YC,YP,PY] be the authority rates vector. It is initialized to [0.3,0.3,0.3,0.3,0.3,0.3,0.3,0.3]. The ground truth ObjVector is [0.7,0.0,0.2,0.2,0.3,0.3,0.3,0.1]. At each iteration we compute the current UserVector produced by the reformulation and compute the cosine similarity cos(ObjVector,UserVector).

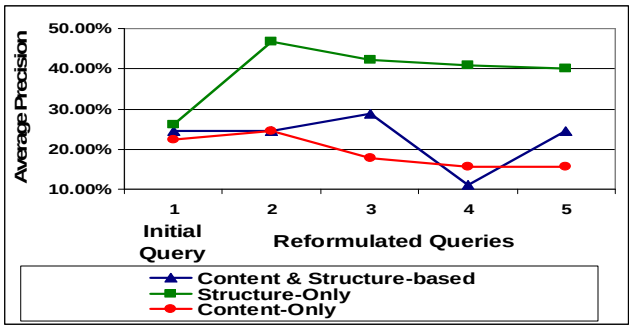


Figure 10: Average Precision for different calibration parameters.

Figure 11 shows the cosine similarity training curves for 4 users averaged over 5 queries each for a different value of C_f (C_e is always 0). We see that the cosine similarity initially increases with the number of iterations and then decreases due to overfitting. Larger C_f values lead to faster peak, since the adjustment of the rates is less smooth (see Equation 13).

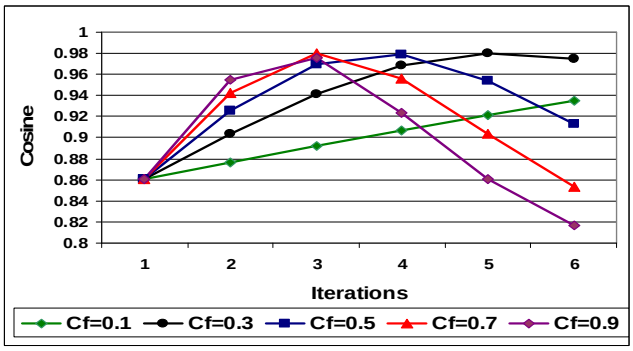


Figure 11: Training of the Authority Transfer Rates.

ObjectRank2 vs. ObjectRank: Finally, we also conducted a survey comparing the quality of ObjectRank2 with ObjectRank [BHP04]. Users were presented with 7 queries (single as well as multi-keyword queries) with Top-10 results obtained using ObjectRank2 and a modified version of ObjectRank as explained below.

Table 2 shows the precision of Top-10 results using ObjectRank2 and ObjectRank techniques. ObjectRank2 is slightly better than ObjectRank for the DBLP dataset, but we expect that it will be considerably better for datasets with longer text descriptions (DBLP titles are too short for IR functions to show a great benefit). We used a slightly modified version of ObjectRank in order to make the

DBLP keyword Queries	*Precision for each query*	
	ObjectRank2	*ObjectRank*
[olap], [query, optimization]	10	9
[xml], [mining], [proximity, search]	10	10
[xml, indexing]	9	8
[ranked, search]	9	10
Average precision	7.7	7.5

Table 2: ObjectRank2 vs. ObjectRank.

comparison fair, with our weighted base set approach. A drawback of ObjectRank [BHP04] is that it favors the more popular keywords in a query. The ObjectRank values of objects tend to be skewed towards popular keywords in a query. The modified formula is given below:

$$r^{t_1,\ldots,t_2}(v) = \prod_{i=1,\ldots,m} (r^{t_i}(v))^{g(t_i)} \qquad (16)$$

where $g(t_i)$ is a *normalizing exponent*, set to $g(t_i) = 1/\log(|S(t_i)|)$.

6.1.2. External Survey.

We conducted an external survey operating on DBLPtop using only structure-based reformulation as it was found to be the best, in the internal survey. Figure 12 shows the average precision curve for 5 iterations averaged over 20 queries by 10 users (2 queries per user). Figure 13 shows the authority transfer rate training curves for the external survey which are similar to those in the internal survey.

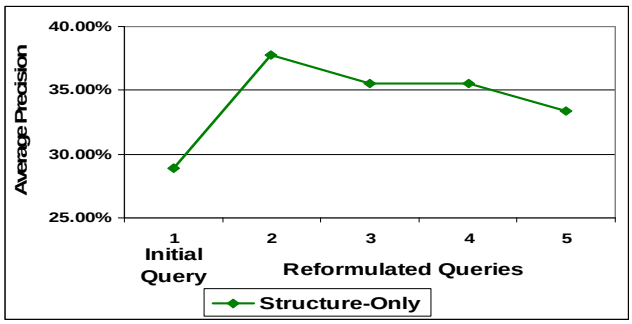


Figure 12: Average Precision using structure-only reformulation with C_f=0.5.

6.2 Performance Experiments

To evaluate the performance of our algorithms, we conducted experiments on all the four datasets. We used a linux machine with Power 4+ 1.7GHz processor and 20GB of RAM. The total execution time is measured for various stages: (a) computing the top-*k* objects for the initial or reformulated query, (b) creating the explaining subgraph, (c) executing the explaining ObjectRank2 on the explaining subgraph, and (d) creating the reformulated query.

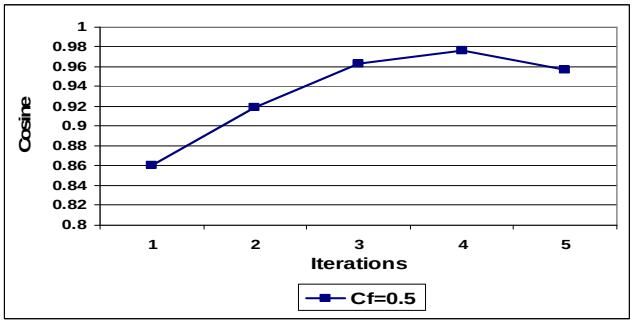
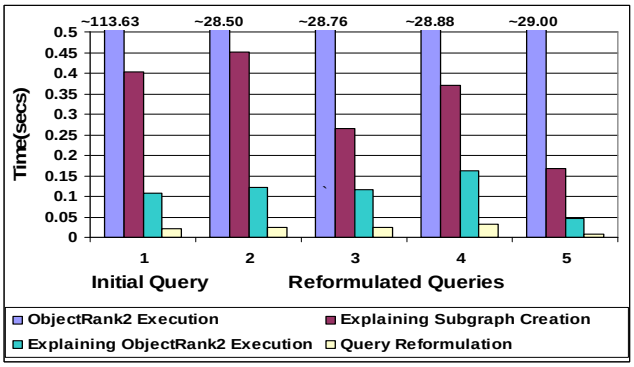


Figure 13: Training of the Authority Transfer Rates.

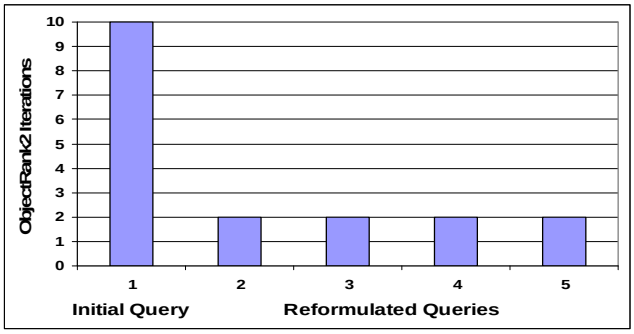
Manipulating Initial ObjectRank values: As in [BHP04], for the initial user query, we initialize every node in D^A with their global ObjectRank values, to achieve faster convergence. Then, for the first reformulated query we use the ObjectRank values of the initial query and so on. The intuition is that the ObjectRank values of the newly reformulated query are expected to be close to the ones obtained by the previous query.

Figures 14(a) and 15(a) show the query and reformulation times over DBLPcomplete and DBLPtop respectively. They show the execution times for the various components of the process: execute the query (first bar), and create the reformulated query (last three bars) at each user feedback and reformulation iteration. We use *L=3* as the radius of the explaining subgraph, and convergence threshold 0.0001. Figures 14(b) and 15(b) show the number of ObjectRank2 iterations for the initial and the reformulated queries over the whole graph. Clearly, using the previous scores as initial values accelerates the convergence of ObjectRank2.



(a): Query and Reformulation Times.

The query reformulation step, where the reformulated query is generated given the converged Explaining ObjectRank2 values, is fast, as the complexity of that step is O(|V|) (in case of content-only reformulation) and O(|E|) (for structure-only reformulation) and O(|V|+|E|) for both content and structure-based reformulations, where V and E are the vertex and edge sets of the explaining subgraph respectively.



(b): ObjectRank2 iterations.

Figure 14: DBLPcomplete Execution.

The ObjectRank2 execution times for DBLPcomplete and DS7 are clearly too long for exploratory searching. This can be addressed in one of the following ways: use faster hardware, precompute ObjectRank2 values as in [BHP04], or define focused subsets like DBLPtop and DS7cancer. The ObjectRank2 execution times for these datasets are 2 seconds for the initial query and less than 1 sec for the subsequent reformulated queries [VHR07].

Table 3 shows the number of Explaining ObjectRank2 (not to be confused with ObjectRank2) iterations over various datasets and for various iterations.

Dataset	Iterations				
	1	2	3	4	5
DBLPcomplete	7.2	8.4	7.4	11	8.4
DBLPtop	7.4	8.2	7.4	8.4	8.6
DS7	5.0	4.8	4.6	5.2	5.6
DS7cancer	4.4	3.8	5.8	5.6	5.0

Table 3: Average Explaining ObjectRank2 Iterations.

A similar set of results are obtained for the biological datasets, DS7 and DS7Cancer as shown in Figures 16 and 17.

7. 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 relevance feedback and query reformulation.

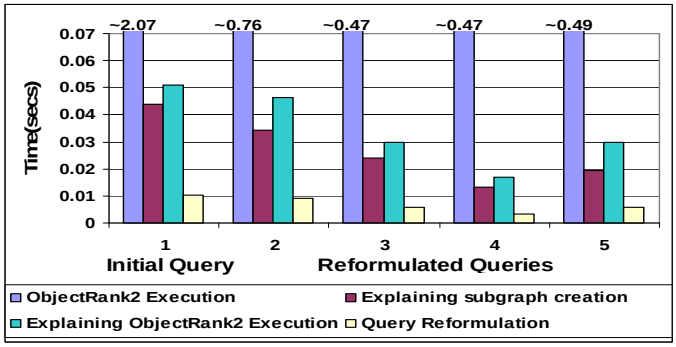
Traditional IR ranking. For an overview of modern IR techniques we refer to [Sin01]. Any state-of-the-art IR ranking function is based on the *tf-idf* principle [Sin01]. The shortcoming of these semantics is that they miss objects that are much related to the keywords, although they do not contain them. The most popular specificity metric in Information Retrieval is the document length (dl). The relevance information is hidden in the link structure of the data graph which is largely ignored by the traditional IR techniques.

Link-based semantics. Savoy [Sav92] was the first to use the link-structure of the Web to discover relevant pages. This idea became more popular with PageRank [BP98], where a global score is assigned to each Web page. HITS [Kle99] employ mutually dependant computation of two values for each web page: hub value and authority. Balmin et al. [BHP04] introduce the ObjectRank metric. 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.

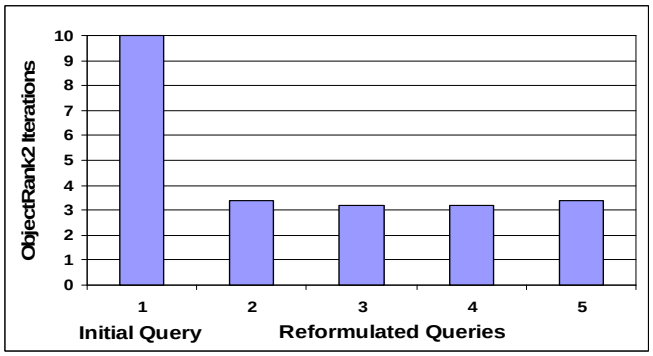
Haveliwala [Hav02] proposes a topic-sensitive PageRank, where the topic-specific PageRanks for each page are precomputed and the PageRank value of the most relevant topic is used for each query. Both works apply to the Web and do not address the unique characteristics of structured databases, as we discuss 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 [GSB+03, HXY03]. XRANK [GSB+03] proposes a way to rank XML elements using the link structure of the database. Furthermore, they introduce a notion similar to ObjectRank transfer edge bounds, to distinguish between containment and IDREF edges. Huang et al. [HXY03] propose a way to rank the tuples of a relational database using PageRank, where connections are determined dynamically by the 33 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.

Relevance Feedback & Query Expansion. Salton and Buckley [SB90] introduced the idea of using relevance feedback for improving search performance. Relevance feedback covers a range of techniques intended to improve a user's query and facilitate retrieval of information relevant to a user's information need. In [BSA94, BSA+95], they showed that query expansion and query term reweighting are essential to Relevance Feedback. For a detailed survey of relevance feedback methods we refer to [RL03, Har92]. The basic approach of term selection, term reweighing and query expansion [Efth93, Har88, MSB98, SVR83, SB95, KF06, XC96, LJ01, HC93] using terms drawn from the relevant documents works well for traditional IR which is content-based. For link-based metrics like ObjectRank [BHP04] this yields poor results as shown in Section 6.1.1. Hence, we need link-based

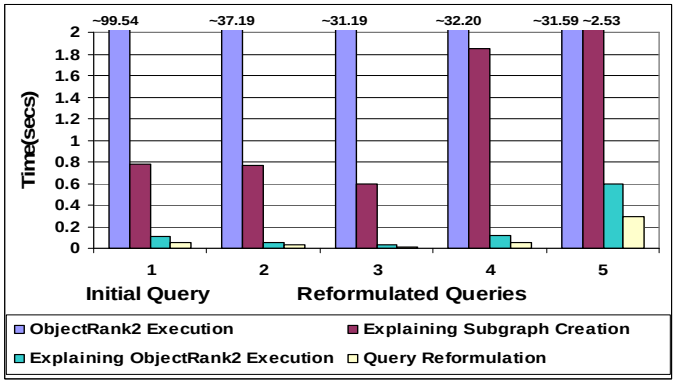


(a): Query and Reformulation Times.

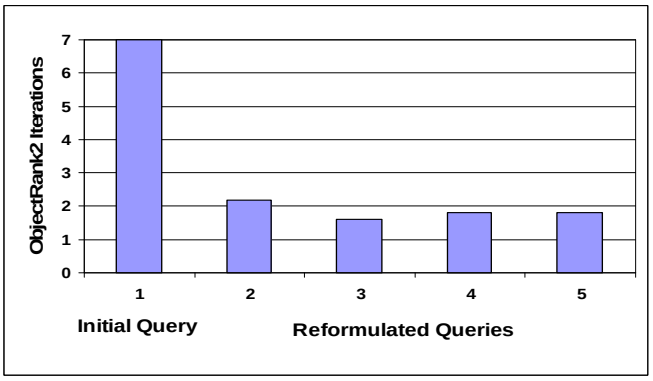


(b): ObjectRank2 iterations.

Figure 15: DBLPtop Execution.

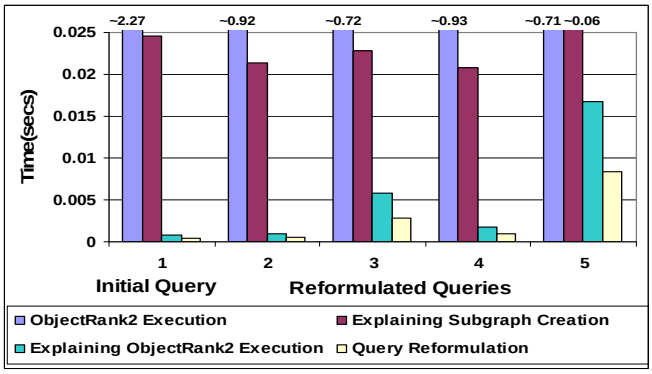


(a): Query and Reformulation Times.

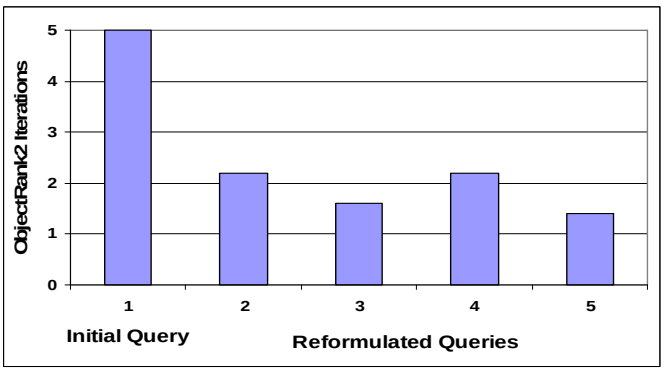


(b): ObjectRank2 iterations.

Figure 16: DS7 Execution.



(a): Query and Reformulation Times.



(b): ObjectRank2 iterations.

Figure 17: DS7cancer Execution.

(structure-based) relevance feedback methods as described in Section 5. A recent work [VB06] on relevance feedback is based on web-graph distance metrics. The basic idea, which is similar to our content-based reformulation technique, is that relevant pages tend to point to other relevance pages, while irrelevant pages are pointed to by other irrelevant pages. Another recent study on relevance propagation over the web [QLZ+05] propose site-based propagation models that out-perform hyperlink-based models. Another recent work [SZ05] describes active feedback algorithms that help to choose documents for relevance feedback so that the system can learn most from the feedback.

8. CONCLUSIONS

In this work we presented a technique to explain the results of authority flow queries and also reformulate them. We discussed reformulations based on content and structure of the underlying graph. We also showed how to automatically train the authority transfer rates of the schema graph based on user preferences. We presented efficient algorithms to

explain and reformulate authority flow queries. We also conducted user surveys to measure the effectiveness of the proposed algorithms. Furthermore, we showed the feasibility of our approach by conducting performance experiments over large graphs.

9. REFERENCES

[ACD02] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A System For Keyword-Based Search Over Relational Databases. ICDE, 2002.

[BHP04] A. Balmin, V. Hristidis, Y. Papakonstantinou: Authority-Based Keyword Queries in Databases using ObjectRank. VLDB 2004.

[BP98] S. Brin and L. Page: The Anatomy of a Large-Scale Hypertextual Web Search Engine. WWW Conference, 1998.

[BSA+95] C. Buckley, G. Salton, J. Allan and A. Singhal. Automatic query expansion using SMART. TREC-3. NIST special publication 500-225. pp 69-80. 1995.

[BSA94] C. Buckley, G. Salton and J. Allan. The effect of adding relevance information in a relevance feedback environment. ACM SIGIR 1994.

[CQ69] A. Collins and M, Quillian, Retrieval Time From Semantzc Memory. J. of Verbal Learning and Verbal Behaviour, Vol 8, pp 240-247, 1969.

[Efth93] E. N. Efthimiadis. Interactive query expansion: A user-centered evaluation of ranking algorithms for interactive query expansion. ACM SIGIR. pp 146-159. 1993.

[GSB+03] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents. ACM SIGMOD, 2003.

[HC93] D. Haines and W.B. Croft. Relevance feedback and inference networks. ACM SIGIR 1993.

[Har88] D. Harman. Towards interactive query expansion. ACM SIGIR pp 321-331. Grenoble. 1988.

[Har92] D. Harman. Relevance feedback and other query modification techniques. Information retrieval: data structures and algorithms, Prentice-Hall Inc, 1992.

[Hav02] T. Haveliwala. Topic-Sensitive PageRank. WWW Conference, 2002.

[HP02] V. Hristidis and Y. Papakonstantinou. DISCOVER: Keyword Search in Relational Databases. VLDB, 2002.

[HPB03] V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword Proximity Search on XML Graphs. ICDE, 2003.

[HXY03] A. Huang, Q. Xue, and J. Yang. TupleRank and Implicit Relationship Discovery in Relational Databases. WAIM, 2003.

[HVP06] H. Hwang, V. Hristidis, and Y. Papakonstantinou. ObjectRank: A System for Authority-based Search on Databases. Demo at SIGMOD, 2006.

[KF06] D. Kelly and X. Fu. Elicitation of term relevance feedback: an investigation of term source and context. ACM SIGIR 2006.

[Kle99] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. Journal of the ACM 46, 1999.

[LJ01] A. M. Lam-Adesina and G.J.F. Jones. Applying summarization techniques for term selection in relevance feedback. ACM SIGIR 2001.

[MSB98] M. Mitra, A. Singhal and C. Buckley. Improving automatic query expansion. ACM SIGIR pp 206-214. 1998.

[QLZ+05] T.Qin, T. Liu, X. Zhang, Z. Chen and W.A study of relevance propagation for web search. ACM SIGIR 2005.

[RWL+06] L. Raschid, Y. Wu, W. Lee, M. E. Vidal, P. Tsaparas, P. Srinivasan, and A. K. Sehgal. Ranking target objects of navigational queries. WIDM 2006.

[RW94] S. E. Robertson and S Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. SIGIR 1994.

[RL03] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems . The Knowledge Engineering Review. 94-145. vol 18, issue 2. 2003.

[SB90] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science. 41. 4. pp 288-297. 1990.

[SB95] G. Salton and C. Buckley. Optimization of relevance feedback weights. ACM SIGIR 1995.

[Sav92] J. Savoy. Bayesian inference networks and spreading activation in hypertext systems. Information Processing and Management, 28(3):389–406, 1992.

[SIY06] P. Shafer, T. Isganitis, G. Yona. Hubs of knowledge: using the functional link structure in Biozon to mine for biologically significant entities. BMC Bioinformatics. 2006 Feb 15;7:71.

[SZ05] X. Shen and C. Zhai. Active feedback in ad hoc information retrieval. ACM SIGIR 2005.

[Sin01] A. Singhal: Modern Information Retrieval: A Brief Overview, Google, IEEE Data Eng. Bull, 2001.

[SVR83] A. Smeaton and C. J. van Rijsbergen. The retrieval effects of query expansion on a feedback document retrieval system. The Computer Journal. 26. 3. pp 239-246. 1983.

[VHR07] R. Varadarajan, V. Hristidis, L. Raschid. Explaining and Reformulating Authority Flow Queries (extended version). http://dbir.cs.fiu.edu/ObjectRankReformulation/, 2007.

[VB06] S. Vassilvitskii and E. Bill. Using web-graph distance for relevance feedback in web search. ACM SIGIR 2006.

[XC96] J.Xu and W.B. Croft. Query expansion using local and global document analysis. ACM SIGIR 1996.