Personalizing PageRank-Based Ranking over Distributed Collections

Stefania Costache, Wolfgang Nejdl, and Raluca Paiu

L3S Research Center / University of Hanover
Deutscher Pavillon, Expo Plaza 1
30539 Hanover, Germany
{costache,nejdl,paiu}@l3s.de

Abstract. In distributed work environments, where users are sharing and searching resources, ensuring an appropriate ranking at remote peers is a key problem. While this issue has been investigated for federated libraries, where the exchange of collection specific information suffices to enable homogeneous TFxIDF rankings across the participating collections, no solutions are known for PageRank-based ranking schemes, important for personalized retrieval on the desktop.

Connected users share fulltext resources and metadata expressing information about them and connecting them. Based on which information is shared or private, we propose several algorithms for computing personalized PageRank-based rankings for these connected peers. We discuss which information is needed for the ranking computation and how PageRank values can be estimated in case of incomplete information. We analyze the performance of our algorithms through a set of experiments, and conclude with suggestions for choosing among these algorithms.

Keywords: PageRank, distributed search, personalization, privacy.

1 Introduction

Collaborative work has become a key factor on the way to success in every company - people do not work isolated, but rather interact with each other by exchanging information, using tools like email clients, IM, blogs, wikis or shared repositories. Every personal desktop thus becomes the sum of all other desktops it interacts with. Accessing these connected information sources in such a collaborative work environment becomes a crucial functionality, which so far has only been partially tackled.

Personal information management [9,10] is a subject of growing interest to the database community, and (distributed and heterogeneous) dataspaces will extend databases beyond centralized and structured information repositories [11]. The *Social Semantic Desktop* paradigm integrates data annotation, organization and search on the desktop, and promises to provide collaborative work environments through connecting all shared data resources in a work group. The

J. Krogstie, A.L. Opdahl, and G. Sindre (Eds.): CAiSE 2007, LNCS 4495, pp. 111–126, 2007.

NEPOMUK[1] project [2] aims to create such an infrastructure, which improves the state of the art in online collaboration and personal data management, by providing seamless access to all information created by single or group efforts.

Peers in the NEPOMUK context share fulltext and semi-structured information, referring to publications, reports and other desktop documents, emails, browsed web pages, address books, etc. These metadata represent additional information about these resources and connect them through semantic relations, such as authorship of papers and reports, sender and recipient information for emails or email attachments. Based on this infrastructure, advanced searching and ranking capabilities can utilize both conventional Information Retrieval (IR)-based information like term frequency in documents and collections, as well as link-related information, the basis of PageRank-like algorithms, e.g., ObjectRank [7,6].

Extending these ranking schemes to a distributed setup is not trivial, because it involves (partial) sharing of possibly private information. Solutions for distributed collections in federated libraries exist, but they provide just traditional IR-based rankings based on TFxIDF metrics through the exchange of collection specific information. We will investigate which resources and information need to be shared to enable personalized PageRank-based ranking among peers, and how algorithms can take privacy constraints for these resources into account. Specifically, we propose and evaluate new algorithms for consistently computing ObjectRank, a PageRank variant appropriate for ranking these connected resources on the desktop.

In Section 2 we will start with the discussion of a search scenario in a distributed work group, and then discuss in detail which information needs to be exchanged in order to achieve appropriate rankings of results. In section 3 we propose and discuss several new algorithms for computing ObjectRank over a set of distributed collections / semantically enabled desktops. In section 4 we describe the experimental setup to evaluate our algorithms, present the experiments we performed and analyze the results. Finally, we discuss related work in section 5 and conclude (section 6).

2 Which Information Should We Exchange?

2.1 A Motivating Scenario

Let's imagine Alice, working in a team with five other students for a research project. Alice's team uses the NEPOMUK-enabled desktop to interact and share information. The team members share papers, project documents and group emails, among others. Papers are annotated with bibliographic information, and connected to the emails they have been attached to. Alice participates in other teams as well, where she shares some of the same documents as well as other information specific only to these other projects. The NEPOMUK infrastructure

[1] This work was supported by the NEPOMUK project funded by the European Commission under the 6th Framework Programme (IST Contract No. 027705).

allows her to search resources on her own desktop as well as on the desktops of her team members, to which Alice's queries are propagated.

The importance of documents (important for the ranking of search results) is influenced by the importance of their authors and conferences, or by the importance of team members sending the document as attachment. These factors are not necessarily the same on each desktop, but are rather based on the conferences relevant to each team member, the number of documents authored by a given person stored on a specific desktop, or the emails connected to these documents. Part of this information (importance of conferences, papers stored on a desktop) can be exchanged easily. Other information such as private emails, or reports from other projects referencing specific papers, should not be exchanged among all participants.

In general, there will be resources that Alice can make public and thus share with everyone, there will be other resources which she will make available only to her trusted friends or to her work mates and there are of course some resources she will never want to share with anybody. This is also true for her contextual metadata generated and stored on her computer, which connects all her resources. Keeping (parts of) her metadata graph private, however, also means that search result rankings at other peers will not be comparable to her own. This unfortunately collides with Alice's desire to get the best ranked matching resources from all her team members connected in her NEPOMUK network (remember that best ranked in this case means "according to Alice's interests / set of resources").

What do we need to exchange in order to provide an appropriate ranking over all document collections Alice asks for results? Clearly, given that the metadata graph determines Alice's ObjectRank scores for all resources (details are described in [7]), we have to exchange PageRank/ObjectRank-related information in addition to the usual IR statistics. We will discuss in the next sections, what can and should be exchanged, in order to rank results for Alice's query on her team members desktops in a way compatible with Alice's ranking. We will take into account the constraint that Alice and her team members do not want to exchange their complete data graphs, which would provide information about all resources they have on their machines.

2.2 Exchanging IR Related Information

Let us first look at a typical scenario in which a user is doing a full-text search over several distributed collections, and wants to rank results according to the usual TFxIDF measures ([4,12]). A query q will consist of several keywords, say q_1 and q_2, and is posed to a broker, which forwards it to a set of m search engines / peers, P'_i, which will then send back to the broker their document rankings R'_i. In practice the user is only interested in the best "top-k" results, where k is usually between 5 and 20. For this, all rankings R'_i have to be merged into one ranked list Rm and the top-k results are presented to the user. Our goal is to achieve the same ranking in the distributed case as produced by the same search on a single collection C containing all documents.

The ranking of the documents in a collection is based on TFxIDF weights, which measure the significance of a word with respect to a document in a collection. The significance of a term increases proportionally to the number of times the term appears in the document, but decreases with the frequency of the term in the whole collection. So, Term Frequency (TF) in the given document gives a measure of importance of the term t_i within that particular document, whereas the Inverted Document Frequency (IDF) is a measure of the general importance of the term. A high weight in TFxIDF is reached by a high TF (in the given document) and a low Document Frequency (DF) of the term in the whole collection of documents.

For distributed retrieval, we want to make the distributed similarity score equal to the similarity scores computed on a single collection C. Therefore, the collection specific values, number of documents (N) and DF, need to be computed before query time (see for example [4]), and recomputed when changes in the collections occur (such as document additions, deletions and updates). To exchange and aggregate them over all collections, we need to send them to the query broker, which can compute the overall Global Inverted Document Frequency ($GIDF$) value, which is then sent back to all search engines. During query execution, all peers will rank results with comparable scores, since they use the common $GIDF$, propagated together with the query. A globally ranked list is achieved by merging the sub-result list entries in descending order of global similarity score. Figure 1 illustrates this process in detail.

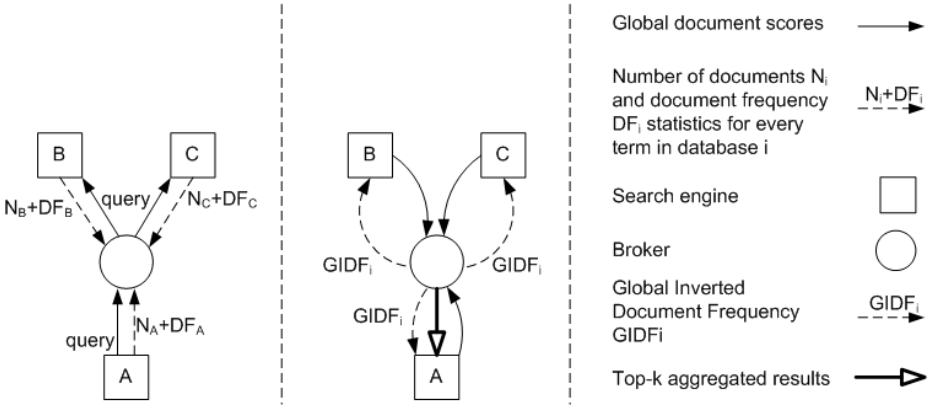

Fig. 1. Statistics propagation for results merging

1. A, B, C send to the *Broker* the total number of documents in the collections (N_A, N_B, N_C) and the DF values. [2]
2. Peer A sends a query to the *Broker* and the *Broker* forwards it to B and C.
3. The *Broker* computes the $GIDF_i$ for each keyword q_i and sends them back to all peers.
4. A, B, C find the matching results for the query and send the top-k results to the *Broker* sorted by the Global Document Scores.
5. The *Broker* merges the results from all peers and sends back to peer A the top-k results.

[2] TF values need not be exchanged since they are document-dependent and therefore do not influence the order of the aggregated result list entries.

2.3 Exchanging ObjectRank Related Information

Let us now look at PageRank / ObjectRank based ranking and which information has to be exchanged to make such rankings on distributed peers compatible with each other. Recall that the computation of PageRank is based on the random surfer model, with the surfer traversing links through the graph of resources, and sometimes jumping randomly to another resource. Then the PageRank value of a resource represents the probability that the random surfer stays on this resource at a given time. If we represent the link structure between all resources through the adjacency matrix A and the random jump through the e vector and the dampening factor d (usually 0.85), PageRank values R are computed through the following eigenvector computation:

$$R = d \cdot A \cdot R + (1 - d) \cdot e \tag{1}$$

For ObjectRank computation, we do not assume the same weight for each link, but rather define link weights based on the type of the connected nodes, through an authority transfer schema [8]. Such a schema specifies how much importance (represented as a real number between 0 and 1) is transferred between connected nodes. The weights of the links between the instances correspond to the weights specified in the authority transfer schema divided by the number of links of the same type. For example, 70% of the importance of a conference node is distributed evenly to each of the publications which are presented at this conference (see [6] for a more detailed description of the algorithm). Let us

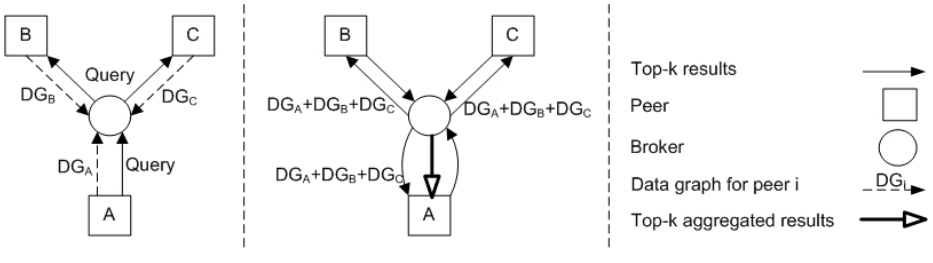

Fig. 2. Aggregated ObjectRank computation

assume, without loss of generality, that all peers use the same authority transfer schema as basis for the ranking computation. Each peer computes ObjectRank scores for its collection. Since this ObjectRank computation is based on the data graph, the adjacency matrix of each peer needs to be updated so that it reflects the new structure created by the integration of the other peers' resources into its own data graph. Therefore, peers need to exchange the URIs of the resources they are sharing, together with the links connecting them. External URIs are integrated into each peer's own data graph of resources. The more resources are shared among peers, the more accurate the aggregated ranked results will be.

Figure 2 presents the necessary steps for computing the aggregated ObjectRank scores in the ideal case, where peers share all resources they own:

1. Peer A sends a query to the *Broker*[3] and the *Broker* forwards it to B and C.
2. The data graph, DG_i is sent to the *Broker* by each peer.
3. The *Broker* merges DG_A+DG_B+DG_C and sends the results to the peers.
4. Peers compute ObjectRank on DG_A+DG_B+DG_C and send top-k results to the *Broker*.
5. The *Broker* merges the results from all peers and sends back to peer A the top-k results.

3 Information Exchange and Rank Computation

3.1 Privacy vs. Information Exchange

The discussion in the previous section assumed the ideal case, where peers share everything they have on their machines. This is usually not the case, instead peers will decide to share only parts of their data graphs and protect the rest. Moreover, peers usually do not want to involve third parties in the exchange process, because this would imply additional privacy and security issues, so they do not want to send data through a broker. We therefore need to develop strategies which do not involve a broker and which allow sending only specific parts of the data graph to the other peers.

As we have already seen, to be able to appropriately rank resources for their neighbors, peers need to know their corresponding data graphs, or at least parts of them. For exchanging this information, peers have the following alternatives:

1. send all nodes in the graph
2. send some of the nodes in the graph
3. send all nodes in the graph, part of them anonymized (the items they want to keep private have hidden URIs, e.g. "hidden_41323")
4. send all nodes in the graph, part of them hashed - which keeps the nodes secret if the other peer does not have them and makes them identifiable if the other peer has them too and uses the same hashing function
5. send all nodes summarized into a world node [5] (which appropriately aggregates node and link information of the graph)

Ranking computation can be based on: a) simple ObjectRank; or b) ObjectRank with biasing [6] on the resources coming from the other peers. We will discuss appropriate combinations of these alternatives in the following.

To describe the graphs used by the different algorithms, we will use the following notations: let $G_i = (V_i, E_i)$ be the data graph of peer i, where V_i and E_i are the corresponding sets of nodes and weighted edges, respectively. In this context, the nodes model the desktop resources (files, emails, visited web pages, etc.), while the edges represent the semantic relationships between them [7]. $G_i' = (V_i', E_i')$ represents the data graph corresponding only to the shared resources, where $G_i' \subset G_i$, $V_i' \subset V_i$, $E_i' \subset E_i$ and $E_i' = \{e_{jk} | j, k \in V_i', j \neq k\}$.

[3] We assume that the peers have already agreed on the authority transfer schema to be used for the ObjectRank computation.

$G_i^{anon} = (V_i^{anon}, E_i^{anon})$ denotes the anonymized data graph of peer i, where $G_i^{anon} = G_i' \cup anonymized(G_i^{unshared})$, $V_i^{anon} = V_i' \cup anonymized(V_i^{unshared})$, $G_i^{unshared} = G_i\ G_i'$ and $E_i^{anon} = E_i$. With $G_i^h = (V_i^h, E_i^h)$ we refer to the hashed data graph, where $G_i^h = hash(G_i)$, $V_i^h = hash(V_i)$ and $E_i^h = E_i$. An example covering all these graphs is presented in figure 3[4].

3.2 Aggregating Graphs into World Nodes

One especially interesting possibility of keeping a graph private, yet provide some information about its connections to the graphs of other peers, is to aggregate all nodes in the graph into a world node and aggregate his connections to the other graphs as well. An example is presented in figure 4, where P2 creates a world node out of its nodes and connects it to the data graph of P1. Using a similar notation as in section 3.1 we define $G_i^{WN} = (V_i^{WN}, E_i^{WN})$, where $V_i^{WN} = WN$ and E_i^{WN} is formed as follows:

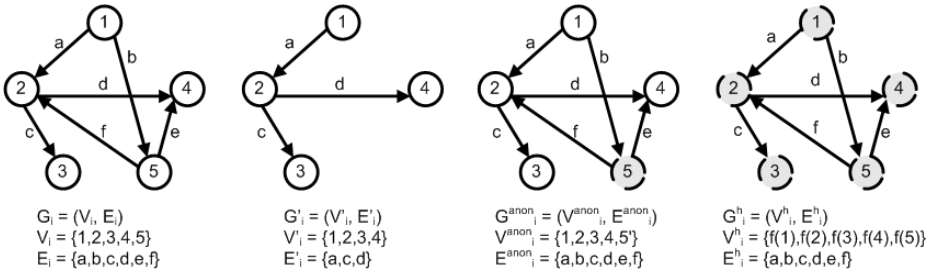

Fig. 3. Example of weighted data graphs - different setups

1. All links from nodes in the other peers' graphs pointing to the nodes in the graph of the peer aggregated into the world node become inlinks of the world node.
2. All links from the nodes of the peer creating the world node pointing to nodes of other peers become outlinks of the world node.
   For a better approximation of the total authority score mass that is received from nodes aggregated in the world node, we weigh every outlink from the world node based on the sum of the weights aggregated into it (the links from the world node to a node of other peers), divided by the number of nodes summarized into the world node.
3. To represent internal links between nodes aggregated into the world node, we create a self-loop link at the world node.
   The weight of this self-loop link is given by the sum of all weights corresponding to the internal links inside the world node, divided by the number of nodes in the world node. The self-loop link represents the probability that a random surfer remains inside the graph that was aggregated into the world node, when following links.

[4] a to f are real numbers, representing the weights of the edges.

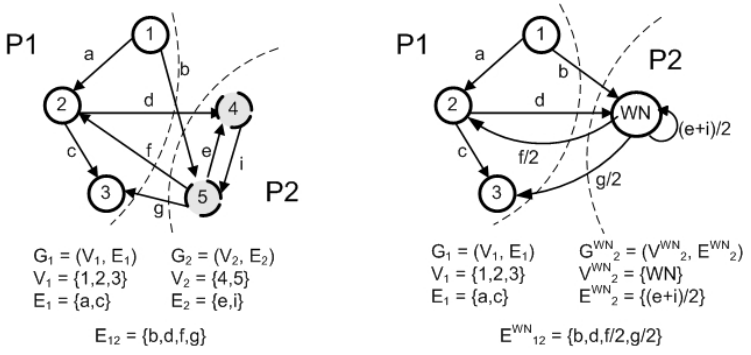

Fig. 4. Example of world node creation

In figure 4 we defined E_{12} as the edges between peers 1 and 2 and E_{12}^{WN} as the edges between P1 and the world node representing P2. An important observation is that for being able to consistently create the world node, a peer needs to know at least a partial structure of the graph of the other peers, otherwise it cannot connect the world node to the other peers' graphs. This means for our setup in figure 4 that P1, who is sending the query, also needs to send its data graph (either the original graph or a hashed version), or at least a part of its graph (original / hashed), such that P2 can correctly put the corresponding inlinks/outlinks to/from its world node.

The big advantage of aggregating everything into a world node is that this protects all internal information about resources and their connections from the receiving peers, while still disclosing (most) information related to external connections and overall weights / scores of the aggregated graph.

3.3 Query Processing and Ranking

Using these notations, we can now distinguish between 8 different query processing and ranking algorithms. These 8 algorithms result as appropriate combinations of the 5 possibilities of exchanging information with the 2 modalities of ranking computation (section 3.1). We eliminated several cases as they proved to be equivalent to the remaining 8 ones. We will describe our algorithms in the following, using 3 peers P_1, P_2 and P_3, with P_1 always sending the query to P2 and P3. In each case P_1 will eventually have a ranked list of results from all peers, including himself.

Algorithm 1 represents the ideal setup, where everything is shared among the three peers, so that each of them can access the aggregated data graph (all peers' graphs merged into one). **Algorithm 2** describes the situation when P1 shares all its resources, but P2 and P3 share only some parts of their data items and anonymize the rest. So P2 and P3 will have complete information regarding P1's graph, but P1 will not know the exact data structures of P2 and P3.

Algorithm 1.

1: P_1 sends G_1 to P_2 and P_3
2: P_2 sends G_2 to P_1 and P_3
3: P_3 sends G_3 to P_1 and P_2
4: Peers aggregate $G_a = G_1 \cup G_2 \cup G_3$
5: Peers compute ObjectRank on G_a

Algorithm 2.

1: P_1 sends G_1 to P_2 and P_3
2: P_2 computes ObjectRank on $G2 = G_1 \cup G_2$
P_3 computes ObjectRank on $G3 = G_1 \cup G_3$
3: P_2 sends G_2^{anon} to P_1
P_3 sends G_3^{anon} to P_1
4: P_1 aggregates $G_a = G_1 \cup G_2^{anon} \cup G_3^{anon}$
5: P_1 computes ObjectRank on G_a

Algorithm 3.

1: P_1 sends G_1^{anon} to P_2 and P_3
2: P_2 computes ObjectRank on
$G2 = G_1^{anon} \cup G_2$
P_3 computes ObjectRank on
$G3 = G_1^{anon} \cup G_3$
3: P_2 sends G_2^{anon} and $R_2 = rank(G2)$ to P_1
P_3 sends G_3^{anon} and $R_3 = rank(G3)$ to P_1
4: P_1 aggregates $G_a = G_1 \cup G_2^{anon} \cup G_3^{anon}$
5: P_1 computes ObjectRank on G_a, biasing on R_2 and R_3

Algorithm 4.

1: P_1 sends G_1' to P_2 and P_3
2: P_2 computes ObjectRank on $G2 = G_1' \cup G_2$
P_3 computes ObjectRank on $G3 = G_1' \cup G_3$
3: P_2 sends G_2^{anon} and $R_2 = rank(G2)$ to P_1
P_3 sends G_3^{anon} and $R_3 = rank(G3)$ to P_1
4: P_1 aggregates $G_a = G_1 \cup G_2^{anon} \cup G_3^{anon}$
5: P_1 computes ObjectRank on G_a, biasing on R_2 and R_3

Algorithm 5.

1: P_1 sends G_1' to P_2 and P_3
2: P_2 computes ObjectRank on $G2 = G_1' \cup G_2$
P_3 computes ObjectRank on $G3 = G_1' \cup G_3$
P_2 and P_3 bias on resources from P_1
3: P_2 sends G_2^{anon} and $R_2 = rank(G2)$ to P_1
P_3 sends G_3^{anon} and $R_3 = rank(G3)$ to P_1
4: P_1 aggregates $G_a = G_1 \cup G_2^{anon} \cup G_3^{anon}$
5: P_1 computes ObjectRank on G_a, biasing on R_2 and R_3

Algorithm 6.

1: P_1 sends G_1' to P_2 and P_3
2: P_2 computes ObjectRank on $G2 = G_1' \cup G_2$
P_3 computes ObjectRank on $G3 = G_1' \cup G_3$
P_2 and P_3 bias on resources from P_1
3: P_2 sends G_2' and $R_2 = rank(G2)$ to P_1
P_3 sends G_3' and $R_3 = rank(G3)$ to P_1
4: P_1 aggregates $G_a = G_1 \cup G_2' \cup G_3'$
5: P_1 computes ObjectRank on G_a, biasing on R_2 and R_3

Algorithm 7.

1: P_1 sends G_1 to P_2 and P_3
2: P_2 computes ObjectRank on $G2 = G_1 \cup G_2$
P_3 computes ObjectRank on $G3 = G_1 \cup G_3$
3: P_2 sends G_2^{WN} and E_{12}^{WN} to P_1
P_2 sends ranked results matching the query
P_3 sends G_3^{WN} and E_{13}^{WN} to P_1
P_3 sends ranked results matching the query
4: P_1 aggregates $G_a = G_1 \cup G_2^{WN} \cup G_3^{WN}$
5: P_1 adds to G_a the edges from $E_{12}^{WN} \cup E_{13}^{WN}$
6: P_1 computes ObjectRank on G_a
P_1 merges P2 and P3 results into final list

Algorithm 8.

1: P_1 sends G_1' to P_2 and P_3
2: P_2 computes ObjectRank on $G2 = G_1' \cup G_2$
P_3 computes ObjectRank on $G3 = G_1' \cup G_3$
3: P_2 sends G_2^{WN} and E_{12}^{WN} to P_1
P_2 sends ranked results matching the query
P_3 sends G_3^{WN} and E_{13}^{WN} to P_1
P_3 sends ranked results matching the query
4: P_1 aggregates $G_a = G_1 \cup G_2^{WN} \cup G_3^{WN}$
5: P_1 adds to G_a the edges from $E_{12}^{WN} \cup E_{13}^{WN}$
6: P_1 computes ObjectRank on G_a
P_1 merges P2 and P3 results into final list

We can also bias ranking computation at P1 on the graphs sent by P2 and P3. In **Algorithm 3**, P1, P2 and P3 share only parts of their resources and anonymize their corresponding data graphs for the items they want to keep private. P2 and P3 compute ObjectRank on the data graph resulting from merging the anonymized data graph of P1 and their own data graph. Results are sent back to P1, which computes ObjectRank on the graph including its own data graph and the anonymized graphs of P2 and P3, biasing the computation on the results coming from P2 and P3. **Algorithm 4**, with P1 sending a subgraph containing only the resources it wants to share, is similar to Algorithm 3.

We can also bias ranking computation at P2 and P3 on the resources received from P1, and then get **Algorithm 5**, based on Algorithm 3, and **Algorithm 6**, based on Algorithm 4. Note that when peers send hashed data graphs, the results will not differ from the case where they anonymize nodes in the private part of their graph. This is because for hashed resources, the receiving peers can identify all resources they share with the sending peers if they use the same hashing function. For the resources they do not share, they will get all information about the link structure, but with the node names unknown / anonymized.

Algorithms 7 and **8** represent the situations where P2 and P3 protect their resources as much as possible, while still providing useful information to P1 using world node aggregation. **Algorithm 7** is a special case of Algorithm 2: P1 shares all its resources but P2 and P3 aggregate their graphs into a world node, keeping the connections to and from P1's graph. **Algorithm 8** is similar to Algorithm 7, only that P1 sends only part of his graph to P2 and P3. In both algorithms, P1 will have to merge results received from P2 and P3 with its own resources, and still keep the relative importance of the items it received, which it can estimate through the information transmitted from P2 and P3 in form of their world nodes, connected to the graph of P1.

All the algorithms we presented can be obviously extended to the general case where a peer is querying in a larger network with more than 2 neighbours.

4 Experiments

4.1 Experimental Setup

To evaluate our algorithms, we gathered metadata from 9 different users (a total of 46500 RDF triples) and partitioned them into 3 sets, the 3 peers. Metadata were produced by a number of metadata generators integrated in Beagle++ [1], and correspond to several types of resources: files, web pages, emails, attachments, publications, persons and conferences. The data set from a single user did not get partitioned into different peers, since we wanted to simulate real peers, with their own profile, but metadata from some of the physical users was copied to more than one peer to simulate different sizes of overlap between the peers. In all considered scenarios, our peers have a common set of data, as we are dealing with peers collaborating with each other. Figure 5 gives an overview: a) resources residing in X are common to all peers; b) slice R contains resources appearing only at peer 1; c) slice O contains resources only from peer 2 and d) slice T contains private resources of peer 3. Based on the amount and type of resources the three peers are sharing, we have three different setups:

1. P1, P2 and P3 share everything, except of some items they want to protect from the uncommon parts, T, O and R;
2. P1, P2, P3 protect resources which can be located both in the common part X, as well as in the uncommon parts of the graph, T, O and R;
3. We experimented with different sizes of the common part X, i.e. the overlap among the peers: a) small; b) medium; and c) large.

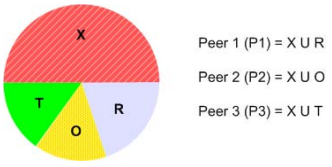

Fig. 5. Peers' resource distribution

For SETUPs 1 and 2 we used **Partitioning 1**, having P1 with 40264 triples, P2 with 7700, P3 with 1786 and a size of the overlap of 1624 triples. For SETUP 3 (**Partitioning 2**) we used a different partitioning: for the big overlap case we divided the set into 45512, 45434, 45584 triples for P1, P2 and P3 respectively and 45015 triples the size of the overlap; for medium overlap 6815 (P1), 44715 (P2), 7120 (P3) and 6075 triples the overlap. The small overlap was simulated with a partitioning of 1215 (P1), 6785 (P2), 38780 (P3) and 140 common triples.

In all our algorithms P1 initiates the query, thus we observe the rank evolution for P1. For all three setups and each algorithm described in section 3, we investigated how the scores of the resources evolve. We compared the ObjectRank scores using 2 similarity metrics between the ObjectRank scores obtained in different algorithms and the ideal case for P1, defined as follows (see also [13]):

1. **OSim** indicates the degree of overlap between the top n elements of two ranked lists τ_1 and τ_2. It is defined as

$$\frac{|Top_n(\tau_1) \cap Top_n(\tau_2)|}{n} \tag{2}$$

2. **KSim** is a variant of Kendall's τ distance measure. Unlike OSim, it measures the *degree of agreement* between the two ranked lists. If U is the union of items in τ_1 and τ_2 and δ_1 is $U \setminus \tau_1$, then let τ_1' be the extension of τ_1 containing δ_1 apearing after all items in τ_1. Similarly, τ_2' is defined as an extension of τ_2. Using these notations, KSim is defined as follows:

$$KSim(\tau_1, \tau_2) = \frac{|(u,v) : \begin{matrix}\tau_1' \text{ and } \tau_2' \text{ agree on order} \\ \text{(u,v), and } u \neq v\end{matrix}|}{|U| \cdot |U-1|} \tag{3}$$

4.2 Results and Analysis

For all three setups we computed KSim and OSim measures (tables 1-5), comparing the ObjectRank results we obtained for algorithms 2-6/2-8 (column 2) against algorithm 1 (column 1), representing the ideal situation, where all peers share everything they have. We analyzed the top 5, 10, 20, 50 and 100[5] ranked results for each algorithm.

[5] ObjectRank is not query dependent, which means that the rankings for specific queries will be a combination between the ObjectRank values and TFxIDF and therefore the matching results can be located beyond top-20.

Table 1. SETUP 1 - OSim, KSim

SETUP 1											
Vs.		Top 5		Top 10		Top 20		Top 50		Top 100	
Algorithm	Algorithm	OSim	KSim	OSim	KSim	OSim	KSim	OSim	KSim	OSim	KSim
1	2	**1.0**	**1.0**	**0.9**	**0.927**	**1.0**	**0.926**	**1.0**	**0.977**	**1.0**	**0.991**
1	3	0.4	0.607	0.6	0.582	0.9	0.670	1.0	0.909	0.96	0.936
1	4	0.4	0.607	0.6	0.582	0.9	0.670	1.0	0.909	0.96	0.936
1	5	0.4	0.607	0.4	0.5	0.55	0.586	0.98	0.805	0.94	0.897
1	6	0.2	0.472	0.3	0.448	0.55	0.534	0.98	0.755	0.91	0.871

Partitioning 1. In SETUP 1 (Table 1), the peers protect resources located only in the non-shared parts, R, O, or T. Given this restriction and the way the world node is constructed we do not need to perform simulations for algorithms 7 and 8, since they yield the same results as in setup 2[6]. In terms of both KSim and OSim, the second algorithm performs best: P1 integrates into its own data graph the anonymized data graphs of P2 and P3, but since P1 is dominating from the point of the number of triples in the graph, this does not have any significant impact on the final scores of P1. Algorithm 6, when every peer biases on the resources received from the others and when only the subgraphs containing the shared resources are sent through the network, performs worst. The reason is that P1 is dominant and the final result will be too much biased on the shared resources of P1. Algorithms 3 and 4 perform the same, as P1 receives the same data graphs in both algorithms.

SETUP 2 (Table 2) differs from SETUP 1 by the fact that the peers can keep private resources from any parts of the graph, X, R, O, or T. When looking at the top-5 ranked results, algorithm 2 still performs good, but as we increase top-k, algorithm 6 gets considerably better. If we consider a small value for k, then for P1 it is better to send part of its data graph containing only the shared resources rather than anonymizing the graph, because anonymization introduces errors (peers are not able to identify what the anonymized resources represent and therefore can introduce duplicates - the resource itself and its anonymized copy). For algorithm 6 with increasing k, biasing on both P2/P3's and P1's side significantly improves the results. Algorithms 7 and 8, using the world node-based approach, perform best, both in terms of OSim and KSim. Evaluating these last two algorithms is done as follows (remember that the list of results contains all nodes of P1 plus the world nodes representing P2 and P3): We merged into the list of P1 (without the world nodes) the lists that P2 and P3 computed after integrating the resources of P1. The way we construct the world node and determine the weights of its outlinks and of the self-loop link models with high fidelity the internal structure of the original graph. Even if the receiving peers do not know the graph structure residing at the other peers - that is the

[6] In algorithm 7 P1 sends all his graph, so that no anonymization is involved which makes SETUP 1 and SETUP 2 exactly the same. For algorithm 8 in SETUP 2, the resources that P1 does not share from X (common part) will still appear in the graphs of P2 and P3, therefore this setup is the same as SETUP 1.

Table 2. SETUP 2 - OSim, KSim

SETUP 2											
Vs.		Top 5		Top 10		Top 20		Top 50		Top 100	
Algorithm	Algorithm	OSim	KSim	OSim	KSim	OSim	KSim	OSim	KSim	OSim	KSim
1	2	0.8	0.6	0.7	0.705	0.9	0.757	0.88	0.873	0.75	0.827
1	3	0.4	0.607	0.6	0.626	0.9	0.701	0.86	0.855	0.81	0.836
1	4	0.6	0.666	0.6	0.648	0.95	0.647	0.8	0.853	0.74	0.806
1	5	0.4	0.607	0.3	0.573	0.65	0.581	0.86	0.8	0.86	0.835
1	6	0.4	0.607	0.4	0.558	0.65	0.581	0.92	0.796	0.89	0.853
1	7	**1.0**	**0.9**	**0.8**	**0.893**	**1.0**	**0.815**	0.96	**0.923**	**0.94**	**0.929**
1	8	**1.0**	**0.9**	**0.8**	**0.893**	**1.0**	**0.815**	**0.98**	**0.923**	0.93	0.912

Table 3. SETUP 3 - Small Overlap

SETUP 3 - Small Overlap											
Vs.		Top 5		Top 10		Top 20		Top 50		Top 100	
Algorithm	Algorithm	OSim	KSim	OSim	KSim	OSim	KSim	OSim	KSim	OSim	KSim
1	2	**1.0**	0.9	**1.0**	**0.977**	**1.0**	**0.989**	0.84	**0.934**	0.87	0.834
1	3	0.6	0.761	0.7	0.666	0.9	0.744	0.88	0.906	0.8	0.835
1	4	0.4	0.607	0.6	0.582	0.85	0.683	0.88	0.883	0.87	**0.869**
1	5	0.6	0.761	0.7	0.666	0.9	0.740	0.82	0.879	0.86	0.846
1	6	0.6	0.666	0.4	0.616	0.6	0.658	0.86	0.780	**0.9**	0.822
1	7	**1.0**	**1.0**	0.6	0.824	**1.0**	0.7	**0.9**	0.888	0.88	0.841
1	8	**1.0**	**1.0**	0.6	0.824	**1.0**	0.7	**0.9**	0.878	0.85	0.817

peer does not disclose any sensible information - the authority transfer among the peers is captured within this model.

Partitioning 2. In SETUP 3 (Tables 3-5) we experimented with 3 different sizes of the overlap.

If the overlap is small or medium, algorithm 2 still performs best for the top-10 and 20 results. If the overlap is big, algorithm 7 performs best for all top-k we consider, followed by algorithm 8 with really small differences. In this case, world nodes (algorithms 7, 8) are strongly connected to the rest of the graph and can therefore very accurately model the influence of the hidden parts of the graph. When looking at top-5 in all variants, algorithms 7 and 8 are the best ones. Algorithms 3 and 4 now perform differently, the biggest difference being for the top-5 ranked results.

Table 4. SETUP 3 - Medium Overlap

SETUP 3 - Medium Overlap											
Vs.		Top 5		Top 10		Top 20		Top 50		Top 100	
Algorithm	Algorithm	OSim	KSim	OSim	KSim	OSim	KSim	OSim	KSim	OSim	KSim
1	2	**1.0**	0.8	**1.0**	**0.955**	**1.0**	**0.984**	0.88	**0.944**	0.77	0.828
1	3	0.6	0.714	0.3	0.625	0.75	0.623	0.86	0.818	0.82	0.797
1	4	0.6	0.714	0.5	0.628	0.7	0.68	0.84	0.829	0.76	0.814
1	5	0.4	0.642	0.5	0.590	0.75	0.68	0.88	0.801	0.82	0.805
1	6	0.4	0.678	0.5	0.638	0.75	0.686	**0.96**	0.811	**0.89**	0.846
1	7	**1.0**	**1.0**	0.6	0.824	**1.0**	0.736	0.9	0.881	0.88	**0.847**
1	8	**1.0**	**1.0**	0.6	0.824	**1.0**	0.7	0.9	0.878	0.86	0.832

Table 5. SETUP 3 - Big Overlap

SETUP 3 - Big Overlap											
Vs.		Top 5		Top 10		Top 20		Top 50		Top 100	
Algorithm	Algorithm	OSim	KSim	OSim	KSim	OSim	KSim	OSim	KSim	OSim	KSim
1	2	0.8	0.6	0.6	0.692	0.85	0.664	0.86	0.864	0.8	0.818
1	3	0.8	0.866	0.6	0.703	0.95	0.661	0.86	0.865	0.8	0.830
1	4	0.6	0.761	0.5	0.619	0.95	0.628	0.86	0.874	0.81	0.845
1	5	0.8	0.866	0.5	0.704	0.8	0.673	0.94	0.828	0.86	0.881
1	6	0.4	0.678	0.5	0.561	0.6	0.648	0.96	0.779	0.89	0.844
1	7	**1.0**	**1.0**	**0.7**	**0.884**	**1.0**	**0.784**	**1.0**	**0.935**	**0.98**	**0.981**
1	8	**1.0**	0.9	**0.7**	0.846	**1.0**	0.684	**1.0**	0.902	0.92	0.931

5 Related Work

In the last two years researchers have investigated how to compute PageRank in a distributed manner. [15] proposes a distributed search engine framework, in which every web server answers queries over its data, and results from multiple web servers are merged into one ranked list. Each web server constructs a web link graph based on its own pages to compute a Local PageRank vector, then they exchange their inter-server link information and compute a ServerRank vector, which is used to refine their Local PageRank vectors. Similarly, [16] computes SiteRank, based on applying PageRank to the graph of Web sites, i.e., the Web graph at the granularity of Web sites instead of Web pages. Aggregating the rankings from multiple sites produces results similar to the true PageRank scores. Both approaches aim to distribute the PageRank computation using several servers and iterations, such that the computational load is reduced, but still the final scores are similar enough with the ones obtained from a global computation. Our goal is to ensure a personalized view over heterogeneous collections, distributed over several desktops, using exchange of appropriate collection/link information before the computation.

[5] was the first paper to introduce the concept of "world node", to incrementally compute a good approximation of PageRank as links evolve. They identify a small portion of the web graph in the vicinity of changes and model the rest of the Web as a single node in this small graph, onto which they compute a version of PageRank and suitably transfer back the results to the original graph. Building on this work, [14] describes a P2P search engine architecture where peers are autonomous, crawl Web fragments and index them locally, but collaborate for query routing and execution. Each peer computes the PageRank scores for the pages it has in its local index. Peers meet and exchange information, and then recompute their PageRank scores. Their original local graph G is extended by adding a special node W, *world node*, representing all pages in the network that do not belong to *G*. Their algorithm assumes that URLs of pages in the world node are known, only their content is not known (not yet crawled). In our scenario, peers do not know the URIs of the external resources and therefore need to send at least part of their data graph to the other peers so that these can create the world node for them. As our world node is used to keep link and

node information private, no inner structure is known. Moreover, all other approaches perform ranking computation on graphs containing only web pages and hyperlinks, while in our case we have different types of links among the nodes, based on their type and on the desktop ontology.

The idea of how communities influence each other is investigated in [3]. They introduce the interesting notion of "energy" of communities, which they define for subsets of the global graph. A community can be viewed as a set of pages on a given topic and the corresponding energy is a measure of the community's authority. The "energy" concept is also applicable in our case, since we are investigating how peers influence each other through the data they are sharing. However, their formulas assume all information about the graph at one location is known, which is not the case in our scenario. It will be interesting to find suitable formulas for approximating energy level and flow for our scenarios, where we have only partial information about the whole graph.

6 Conclusions

An important functionality in distributed work environments is to provide searching and ranking capabilities over collections distributed over the desktops of a work group. In this paper we introduced several algorithms for retrieving resources over a network of such desktops, which rely on the exchange of collection specific information between the participating peers in order to achieve appropriate ranking using PageRank-based algorithms. All our algorithms take privacy into account, i.e. peers want to exchange only certain parts of their desktop content, a constraint which has been neglected so far in all previous work on distributed PageRank computation.

We analyzed in detail how our algorithms perform in several setups of resource sharing. In particular, we experimented with different sizes of data sets residing on the peers' desktops and with different dimensions of the overlapping information. Our experiments show that we can compute appropriate ObjectRank values even if the peers do not share everything they have. Specifically, algorithms aggregating node and link information into one "world node" proved to be the best tradeoff between privacy and quality. They offer the best way of protecting resources, since peers do not reveal any of their nodes or the way they are interconnected, approximate ObjectRank values very well, and guarantee the smallest network load. In future work we will extend these algorithms with methods to estimate the potential of peers to influence results of other peers, and come up with incremental update schemes when peer content changes.

References

1. Beagle++. (2006) `http://beagle.kbs.uni-hannover.de/`
2. NEPOMUK - The Social Semantic Desktop. (2006) `http://nepomuk.semanticdesktop.org`
3. Bianchini, M., Gori, M., Scarselli, F.: Inside pagerank. ACM Trans. Inter. Tech. 5(1), 92–128 (2005)

4. Callan, J.P., Lu, Z., Croft, W.B.: Searching distributed collections with inference networks. In: Proc. of the Intl. Conf. on Research and Development in Information Retrieval (SIGIR) (1995)
5. Chien, S., Dwork, C., Kumar, S., Sivakumar, D.: Towards exploiting link evolution. In: Unpublished manuscript (2001)
6. Chirita, P.A., Costache, S., Nejdl, W., Paiu, R.: Beagle++: Semantically enhanced searching and ranking on the desktop. In: Proc. of the European Semantic Web Conf. (ESWC) (2006)
7. Chirita, P.A., Ghita, S., Nejdl, W., Paiu, R.: Semantically enhanced searching and ranking on the desktop. In: Proc. of the Semantic Desktop Workshop held at the Intl. Semantic Web Conf (2005)
8. Damian, A., Nejdl, W., Paiu, R.: Peer-sensitive objectrank: Valuing contextual information in social networks. In: Proc. of the Intl. Conf. on Web Information Systems Engineering (2005)
9. Dong, X., Halevy, A.Y.: A platform for personal information management and integration. In: Proc. of Conf. on Innovative Data Systems Research (CIDR) (2005)
10. Dong, X., Halevy, A.Y., Nemes, E., Sigundsson, S.B., Domingos, P.: Semex: Toward on-the-fly personal information integration. In: Proc. of the Workshop on Information Integration on the Web (2004)
11. Franklin, M., Halevy, A.Y., Maier, D.: From databases to dataspaces: a new abstraction for information management. SIGMOD Rec. 34(4), 27–33 (2005)
12. Green, N., Ipeirotis, P.G., Gravano, L.: SDLIP + STARTS = SDARTS a protocol and toolkit for metasearching. In: ACM/IEEE Joint Conference on Digital Libraries, pp. 207–214 (2001)
13. Haveliwala, T.: Topic-sensitive pagerank. In: Proc. of the Intl. WWW Conf (2002)
14. Parreira, J.X., Donato, D., Michel, S., Weikum, G.: Efficient and decentralized pagerank approximation in a peer-to-peer web search network. In: Proc. of the Intl. Conf. on Very Large Data Bases (VLDB) (2006)
15. Wang, Y., DeWitt, D.: Computing pagerank in a distributed internet search system. In: Proc. of the Intl. Conf. on Very Large Databases (VLDB) (2004)
16. Wu, J., Aberer, K.: Using siterank for decentralized computation of web document ranking. In: Proc. of Intl. Conf. on Adaptive Hypermedia and Adaptive WebBased Systems (2004)