

Beagle⁺⁺: Semantically Enhanced Searching and Ranking on the Desktop

Paul - Alexandru Chirita, Stefania Costache, Wolfgang Nejdl, and Raluca Paiu

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

Abstract. Existing desktop search applications, trying to keep up with the rapidly increasing storage capacities of our hard disks, offer an incomplete solution for information retrieval. In this paper we describe our Beagle⁺⁺ desktop search prototype, which enhances conventional full-text search with semantics and ranking modules. This prototype extracts and stores activity-based metadata explicitly as RDF annotations. Our main contributions are extensions we integrate into the Beagle desktop search infrastructure to exploit this additional contextual information for searching and ranking the resources on the desktop. Contextual information plus ranking brings desktop search much closer to the performance of web search engines. Initially disconnected sets of resources on the desktop are connected by our contextual metadata, PageRank derived algorithms allow us to rank these resources appropriately. First experiments investigating precision and recall quality of our search prototype show encouraging improvements over standard search.

1 Introduction

The capacity of our hard-disk drives has increased tremendously over the past decade, and so has the number of files we usually store on our computer. It is no wonder that sometimes we cannot find a document any more, even when we know we saved it somewhere. Ironically, in quite a few of these cases, the document we are looking for can be found faster on the World Wide Web than on our personal computer.

Web search has become more efficient than PC search due to the boom of web search engines and to powerful ranking algorithms like the PageRank algorithm introduced by Google [12]. The recent arrival of desktop search applications, which index all data on a PC, promises to increase search efficiency on the desktop. However, even with these tools, searching through our (relatively small set of) personal documents is currently inferior to searching the (rather vast set of) documents on the web. This happens because these desktop search applications cannot rely on PageRank-like ranking mechanisms, and they also fall short of utilizing desktop specific characteristics, especially context information.

We therefore have to enhance simple indexing of data on our desktop by more sophisticated ranking techniques, otherwise the user has no other choice but to look at the entire result sets for her queries – usually a tedious task. The main problem with ranking on the desktop comes from the lack of links between documents, the foundation of current ranking algorithms (in addition to TF/IDF numbers). A semantic desktop offers the missing ingredients: By gathering semantic information from user activities, from the contexts the user works in¹, we build the necessary links between documents.

We begin this paper by presenting related work in Section 2, continuing with our proposed desktop system architecture described in detail in Section 3. In this paper we enhance and contextualize desktop search based on both resource specific and semantic metadata collected from different available contexts and activities performed on a personal computer. We describe the semantics of these different contexts by appropriate ontologies in Section 3.3, and then propose a ranking algorithm for desktop documents in Section 3.4. For this latter aspect, we focus on recent advances of PageRank-based ranking, showing how local (i.e., context-based) and global ranking measures can be integrated in such an environment. We are implementing our prototype on top of the open source Beagle project [10], which aims to provide basic desktop search in Linux. Section 4 gives a detailed description of our prototype, and shows how we extended Beagle with additional modules for annotating and ranking resources. In Section 5 we investigate the quality of our algorithms in a typical desktop search scenario, and illustrate how our extended Beagle⁺⁺ search infrastructure improves the retrieval of search results in terms of number (recall) and order (precision) of results, using context and ranking based on this information.

2 Previous Work

The difficulty of accessing information on our computers has prompted several first releases of desktop search applications recently. The most prominent examples include Google desktop search [11] (proprietary, for Windows) and the Beagle open source project for Linux [10]. Yet they include *no* metadata whatsoever in their system, but just a regular text-based index. Nor does their competitor MSN Desktop Search [14]. Finally, Apple Inc. promises to integrate an advanced desktop search application (named *Spotlight Search* [2]) into their upcoming operating system, Mac OS Tiger. Even though they also intend to add semantics into their tool, only explicit information is used, such as file size, creator, or metadata embedded into specific files (images taken with digital cameras for example include many additional characteristics, such as exposure information). While this is indeed an improvement over regular search, it still misses contextual information often resulting or inferable from explicit user actions or additional background knowledge, as discussed in the next sections.

¹ Studies have shown that people tend to associate things to certain contexts [17], and this information should be utilized during search. So far, however, neither has this information been collected, nor have there been attempts to use it.

Swoogle [7] is a search and retrieval system for finding semantic web documents on the web. The ranking scheme used in Swoogle uses weights for the different types of relations between Semantic Web Documents (SWD) to model their probability to be explored. However, this mainly serves for ranking between ontologies or instances of ontologies. In our approach we have instances of a fixed ontology and the weights for the links model the user's preferences.

Facilitating search for information the user has already seen before is also the main goal of the *Stuff I've Seen (SIS)* system, presented in [8]. Based on the fact that the user has already seen the information, contextual cues such as time, author, thumbnails and previews can be used to search for and present information. [8] mainly focuses on experiments investigating the general usefulness of this approach though, without presenting more technical details.

The importance of semantically capturing users' interests is analyzed in [1]. The purpose of their research is to develop a ranking technique for the large number of possible semantic associations between the entities of interest for a specific query. They define an ontology for describing the user interest and use this information to compute weights for the links among the semantic entities. In our system, the user's interest is a consequence of her activities, this information is encapsulated in the properties of the entities defined, and the weights for the links are manually defined.

An interesting technique for ranking the results for a query on the semantic web takes into consideration the inferencing processes that led to each result [16]. In this approach, the relevance of the returned results for a query is computed based upon the specificity of the relations (links) used when extracting information from the knowledge base. The calculation of the relevance is however a problem-sensitive decision, and therefore task oriented strategies should be developed for this computation.

3 An Architecture for Searching the Semantic Desktop

3.1 Overview

This chapter will present our 3-layer architecture for generating and exploiting the metadata enhancing desktop resources. At the bottom level, we have the physical resources currently available on the PC desktop. Even though they can all eventually be reduced to files, it is important to differentiate between them based on content and usage context. Thus, we distinguish structured documents, emails, offline web pages, general files² and file hierarchies. Furthermore, while all of them do provide a basis for desktop search, they also miss a lot of contextual information, such as the author of an email or the browsing path followed on a specific web site. We generate and store this additional search input using RDF metadata, which is placed on the second conceptual layer of our architecture. Finally, the uppermost layer implements a ranking mechanism over all resources on the lower levels. An importance score is computed for each desktop item,

² Text files or files whose textual content can be retrieved.

supporting an enhanced ordering of results within desktop search applications. The architecture is depicted in Figure 1. In the next subsections we describe each of its layers following a bottom-up approach.

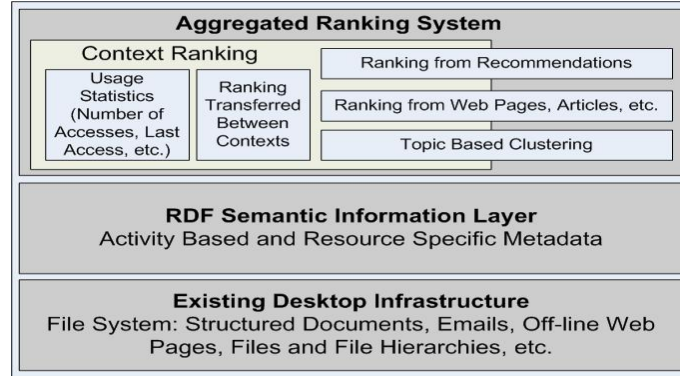


Fig. 1. Desktop Ranking System Architecture

3.2 Current Desktop Infrastructure and its Limitations

Motivation and Overview. Today the number of files on our desktops can easily reach 10,000, 100,000 or more. This large amount of data can no longer be ordered with manual operations such as defining explicit file and directory names. Automatic solutions are needed, preferably taking into account the activity contexts under which each resource was stored/used. In our prototype we focus on three main working contexts of email exchanges, file procedures (i.e., create, modify, etc.), and web surfing. Furthermore, we investigate an additional extended context related to research and scientific publications. In the following paragraphs, we discuss how the resources associated to these contexts are currently encountered, and which (valuable) information is lost during their utilization. Subsequent sections present solutions to represent this information and exploit it for desktop search applications.

Email Context. One of the most flourishing communication medium is surely email communication. International scientific collaboration has become almost unthinkable without electronic mail: Outcomes of brainstorming sessions, intermediate versions of reports, or articles represent just a few of the items exchanged within this environment. Similarly, Internet purchasing or reservations are usually confirmed via email. Considering the continuous increase of email exchanges, enhanced solutions are necessary to sort our correspondence. More, when storing emails, a lot of contextual information is lost. Most significant here is the semantic connection between the attachments of an email, its sender and subject information, as well as the valuable comments inside its body. This

information should be explicitly represented as RDF metadata, to enable both enhanced search capabilities for our inbox, as well as the exploitation of the semantic links between desktop files (e.g., PDF articles stored from attachments), the person that sent them to us and the comments he added in the email body.

Files and File Hierarchy Context and Web Cache Context. Similar to the discussion above, various semantic relationships exist in other contexts such as file and web cache context. Due to space limitations, we refer the reader to [5], where we proposed several solutions to enrich the information associated to file and directory names, as well as to previously visited resources on the Web.

Working with Scientific Publications. Research activities represent one of the occupations where the need for contextual metadata is very high. The most illustrative example is the document itself: Where did this file come from? Did we download it from CiteSeer or did somebody send it to us by email? Which other papers did we download or discuss via email at that time, and how good are they (based on a ranking measure or on their number of citations)? We might remember the general topic of a paper and the person with whom we discussed about it, but not its title. These situations arise rather often in a research environment and have to be addressed by an appropriate search infrastructure.

3.3 RDF Semantic Information Layer

Motivation and Overview. People organize their lives according to preferences often based on their activities. Consequently, desktop resources are also organized according to performed activities and personal profiles. Since, as described above, most the information related to these activities is lost on our current desktops, the goal of the RDF semantic information layer is to record and represent this data and to store it in RDF annotations associated to each resource. Figure 2 depicts an overview image of the ontology that defines appropriate annotation metadata for the context we are focusing on in this paper. The following paragraphs will describe these metadata in more detail.

Email Metadata. Basic properties for the email context refer to the date when an email was sent or accessed, as well as its subject and body text. The status of an email can be described as seen/unseen or read/unread. A property “reply to” represents email thread information, the “has attachment” property describes a 1:n relation between an email and its attachments. The “sender” property gives information about the email sender, which can be associated to a social networking trust scheme, thus providing valuable input for assessing the quality of the email according to the reputation of its sender.

File and Web Cache Specific Metadata. For these, we again refer the reader to our previous work [5] which describes the ontologies associated to these activity contexts. An overview can be found in the lower half of Figure 2.

Scientific Publications Metadata. The Publication class represents a specific type of file, with additional information associated to it. The most common fields are “author”, “conference”, “year”, and “title”, which comprise the regular information describing a scientific article. Additionally, we store the paper’s CiteSeer ID if we have it. The publication context is connected to the email

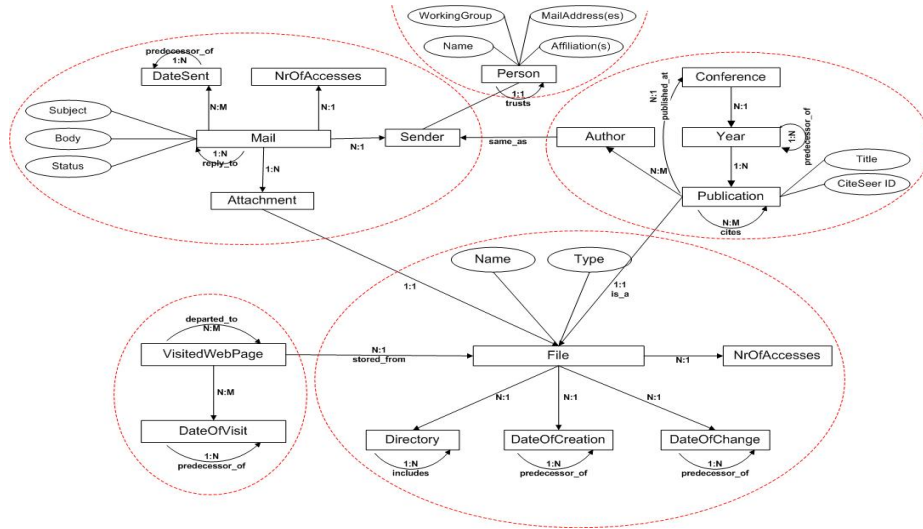


Fig. 2. Contextual Ontology for the Semantic Desktop

context, if we communicate with an author or if we save a publication from an email attachment. Of course, since each publication is stored as a file, it is also connected to the file context, and thus to the file specific information associated to it (e.g., path, number of accesses, etc.).

3.4 Aggregated Ranking System

Motivation and Overview. As the amount of desktop items has been increasing significantly over the past years, desktop search applications will return more and more hits to our queries. Contextual metadata, which provide additional information about each resource, result in even more search results. A measure of importance is therefore necessary, which enables us to rank these results. The following paragraphs describe such a ranking mechanism, based on the Google PageRank algorithm [12].

Basic Ranking. Given the fact that rank computation on the desktop would not be possible without the contextual information, which provides semantic links among resources, annotation ontologies should describe all aspects and relationships among resources influencing the ranking. The identity of the authors for example influences our opinion of documents, and thus “author” should be represented explicitly as a class in our publication ontology.

Second, we have to specify how these aspects influence importance. Object-Rank [4] has introduced the notion of authority transfer schema graphs, which extend schemas similar to the ontologies previously described, by adding weights and edges in order to express how importance propagates among the entities and resources inside the ontology. These weights and edges represent authority transfer annotations, which extend our context ontologies with the information

we need to compute ranks for all instances of the classes defined in the context ontologies.

Figure 3 depicts our context ontology plus appropriate authority transfer annotations. For example, authority of an email is split among the sender of the email, its attachment, the number of times that email was accessed, the date when it was sent and the email to which it was replied. If an email is important, the sender might be an important person, the attachment an important one and/or the number of times the email was accessed is very high. Additionally, the date when the email was sent and the previous email in the thread hierarchy also become important. As suggested in [4], every edge from the schema graph is split into two edges, one for each direction. This is motivated by the observation that authority potentially flows in both directions and not only in the direction that appears in the schema: if we know that a particular person is important, we also want to have all emails we receive from this person ranked higher. The final ObjectRank value for each resource is calculated based on the PageRank formula (presented in Section 4.3).

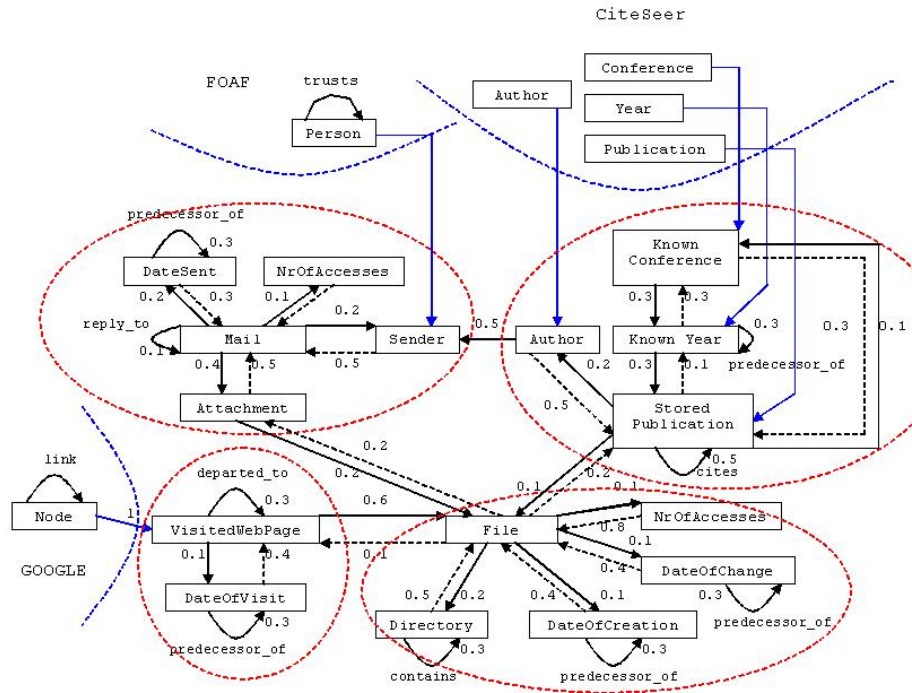


Fig. 3. Contextual Authority Transfer Schema

Using External Sources. For the computation of authority transfer, we can also include additional external ranking sources to connect global ranking

computation and personalized ranking of resources on our desktop. These external ranking sources are used to provide the seed values for the calculation of the personal ranking. Our prototype ontology includes three global ranking services, one returning Google ranks, the second one ranks computed from the CiteSeer database and the last one from the social network described with FOAF.

The ObjectRank value for each resource is calculated based on the PageRank formula and the seed values for this computation integrate information from external ranking systems and personalized information. We use the following external ranking systems as the most relevant for our purpose:

- *Ranking for articles.* Co-citation analysis is used to compute a primary rank for the article [15]. Because of the sparse article graph on each desktop this rank should be retrieved from a server that stores the articles (in our case all metadata from CiteSeer and DBLP).
- *Recommendations.* We may receive documents from other peers together with their recommendations. These recommendations are weighted by a local estimate of the sender’s expertise in the topic [9, 6].

Personalization. Different authority transfer weights express different preferences of the user, translating into personalized ranking. The important requirement for doing this successfully is that we include in a users ontology all concepts, which influence her ranking function. For example, if we consider a publication important because it was written by an author important to us, we have to represent that in our context ontology. Another example are digital photographs, whose importance is usually heavily influenced by the event or the location where they were taken. In this case both event and location have to be included as classes in our context ontology. The user activities that influence the ranking computation have also to be taken into account, which translates to assigning different weights to different contexts.

4 Beagle⁺⁺ Prototype

Our current prototype is built on top of the open source Beagle desktop search infrastructure, which we extended with additional modules: metadata generators, which handle the creation of contextual information around the resources on the desktop, and a ranking module, which computes the ratings of resources so that search results are shown in the order of their importance. The advantage of our system over existing desktop search applications consists in both the ability of identifying resources based on an extended set of attributes – more results, and of presenting the results according to their ranking – to enable the user to quickly locate the most relevant resource.

4.1 Current Beagle Architecture

The main characteristic of our extended desktop search architecture is metadata generation and indexing on-the-fly, triggered by modification events generated

upon occurrence of file system changes. This relies on notification functionalities provided by the kernel. Events are generated whenever a new file is copied to hard disk or stored by the web browser, when a file is deleted or modified, when a new email is read, etc. Much of this basic notification functionality is provided on Linux by an inotify-enabled Linux kernel, which is used by Beagle.

Our Beagle⁺⁺ prototype keeps all the basic structure of Beagle and adds additional modules that are responsible for generating and using the contextual annotations enriching the resources on our desktop. The main components of the extended Beagle prototype are Beagled⁺⁺ and Best⁺⁺, as seen in Figure 4, "++" being used to denote our extensions. Beagled⁺⁺ is the main module that deals with indexing of resources on the desktop and also retrieving the results from user queries. Best⁺⁺ is responsible for the graphical interface, and communicates with Beagled⁺⁺ through sockets. When starting Beagle, the query driver is started and waits for queries. The Best⁺⁺ interface is initialized, too, and is responsible for transmitting queries to Beagled⁺⁺ and visualization of answers.

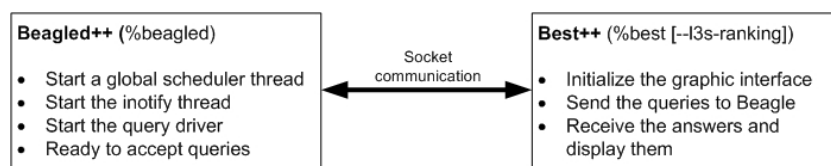


Fig. 4. Extended Beagle Desktop Search

4.2 Extending Beagle with Metadata Generators

Depending on the type and context of the file / event, metadata generation is performed by appropriate metadata generators, as described in Figure 5. These applications build upon an appropriate RDFS ontology as shown in [5], describing the RDF metadata to be used for that specific context. Generated metadata are either extracted directly (e.g. email sender, subject, body) or are generated using appropriate association rules plus possibly some additional background knowledge. All of these metadata are exported in RDF format, and added to a metadata index, which is used by the search application together with the usual full-text index [13].

The architecture of our prototype environment includes four prototype metadata generators according to the types of contexts described in the previous sections. We added a new subclass of the LuceneQueryable class, MetadataQueryable, and, from this one, derived four additional subclasses, dealing with the generation of metadata for the appropriate contexts (Files, Web Cache, Emails and Publications). The annotations we create include the corresponding elements

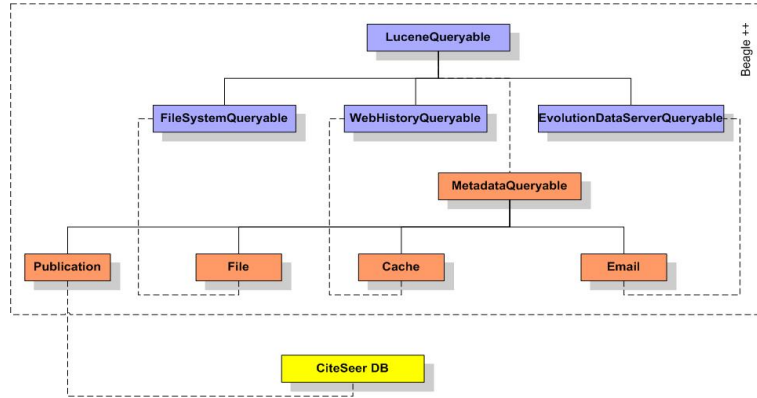


Fig. 5. Beagle Extensions for Metadata Support

depicted in the ontology graph Figure 2. They are described in detail in [5]. A new one is the publication metadata generator, described in the next paragraph.

Publication Metadata Generator. For the experiments described in this paper, we have implemented a metadata generator module which deals with publications. For each identified paper, it extracts the title and tries to match it with an entry into the CiteSeer publications database. If it finds an entry, the application builds up an RDF annotation file, containing information from the database about the title of the paper, the authors, publication year, conference, papers which cite this one and other CiteSeer references to publications. All annotation files corresponding to papers are merged in order to construct the RDF graph of publications existing on one’s desktop.

4.3 Extending Beagle with A Ranking Module

Each user has his own contextual network / context metadata graph and for each node in this network the appropriate ranking as computed by the algorithm described in section 3.4. The computation of rankings is based on the link structure of the resources as specified by the defined ontologies and the corresponding metadata. We base our rank computation on the PageRank formula

$$r = d \cdot A \cdot r + (1 - d) \cdot e \quad (1)$$

applying the random surfer model and including all nodes in the base set. The random jump to an arbitrary resource from the data graph is modeled by the vector e . [6] shows how, by appropriately modifying the e vector, we can take the different trust values for the different peers sending information into account. A is the adjacency matrix which connects all available instances of the existing context ontology on one’s desktop. The weights of the links between the instances correspond to the weights specified in the authority transfer annotation ontology divide by the number of the links of the same type. When instantiating the

authority transfer annotation ontology for the resources existing on the users desktop, the corresponding matrix A will have elements which can be either 0, if there is no edge between the corresponding entities in the data graph, or they have the value of the weight assigned to the edge determined by these entities, in the authority transfer annotation ontology, divided by the number of outgoing links of the same type.

The original Beagle desktop search engine uses the facilities provided by Lucene.NET for ranking the results, which means that Beagle's hits are scored only based on TF/IDF measures. Such a ranking scheme gives good results in the case of documents explicitly containing the keywords in the query. Still, as discussed above, TF/IDF alone is not sufficient, as it does not exploit any additional hints about importance of information.

We have therefore implemented a new ranking scheme in Beagle⁺⁺, which profits from the advantages offered by TF/IDF, but takes into account ObjectRank scores as described in this paper. For all resources existing on the desktop this scheme computes the ranks with our ObjectRank-based algorithm presented above, and the resulting ranks are then combined with the TF/IDF scores provided by Lucene.NET using the following formula:

$$R'(a) = R(a) * TF \times IDF(a), \quad (2)$$

where:

a - represents the resource

$R(a)$ - is the computed ObjectRank

$TF \times IDF(a)$ - is the TF/IDF score for resource a

This formula guaranties that the hits will have a high score if they both have a high ObjectRank and a TF/IDF score.

The user is able to chose one of the two available ranking schemas: the one provided by Beagle, based on TF/IDF measures, or the one we developed, based on ObjectRank plus TF/IDF. The first scheme is implicit. For the second one, users have to specify an additional parameter when starting the Best client: *%best -l3s-ranking*.

5 Experiments

5.1 Experimental Setup

We did a first evaluation of our algorithms by conducting a small scale user study. Colleagues of ours provided a set of their locally indexed publications, some of which they received as attachments to emails (thus containing rich contextual metadata associated to them from the specific email fields). Then, each subject defined her *own* queries, related to their activities, and performed search over the above mentioned reduced images of their desktops. In total, 30 queries were issued. The average query length was 2.17 keywords, which is slightly more than the average of 1.7 keywords reported in other larger scale studies (see for example [8]). Generally, the more specific the test queries are, the more difficult it is

to improve over basic textual information retrieval measures such as TFxIDF. Thus, having an average query length a bit higher than usual can only increase the quality of our conclusions.

For comparison purposes, we sent each of these queries to three systems: (1) the original Beagle system (with output selected and sorted using solely TFxIDF), (2) an intermediate version of Beagle⁺⁺ enhanced only with activity based metadata (using the same TFxIDF measure for ordering its output, but giving more importance to metadata results than to regular desktop items), and (3) the current Beagle⁺⁺, containing enhancements for both metadata support and desktop ranking. For every query and every system, each user rated the top 5 output results using grades from 0 to 1, as follows: 0 for an irrelevant result, 0.5 for a relevant one, and 1 for highly relevant one.

5.2 Methodology

Even when semantic information (e.g., RDF annotations, etc.) is integrated as part of a search system, the traditional measures from information retrieval theory can and should still be applied when evaluating system performance. We therefore used the ratings of our subjects to compute average precision and recall values at each output rank [3]. In general, precision measures the ability of an (information retrieval) system to return *only* relevant results. It is defined as:

$$\text{Precision} = \frac{\text{Number of } \textit{Relevant} \textit{ Returned Results}}{\text{Number of Returned Results}} \quad (3)$$

Recall is its complement: It measures the ability of a system to return *all* relevant documents, and is computed using the formula below:

$$\text{Recall} = \frac{\text{Number of Relevant Returned Results}}{\text{Total Number of Relevant Results Available in the Entire System}} \quad (4)$$

Both measures can be calculated at any rank r , i.e., considering only the top r results output by the application. For example, even if the system has returned 2000 hits for some user query, when calculating precision at the top-3 results, we consider only these three as returned results. This is necessary for large scale environments, such the World Wide Web, and more recently, the PC desktop, because it impossible to check the relevance of all output results – even in the desktop environment, it is not uncommon to obtain several hundreds of search results to a given query. Restricting the calculation of precision and recall to various ranks is also useful in order to investigate the quality of the system at different levels. Usually, in a healthy information retrieval system, as the rank level is increased, recall is also increasing (the denominator remains the same, while the numerator has the possibility to increase), whereas precision is decreasing (because most of the relevant results should be at the very top of the list).

Another important aspect is calculating the total number of available relevant results. For search engines, including desktop ones, an approximation must be

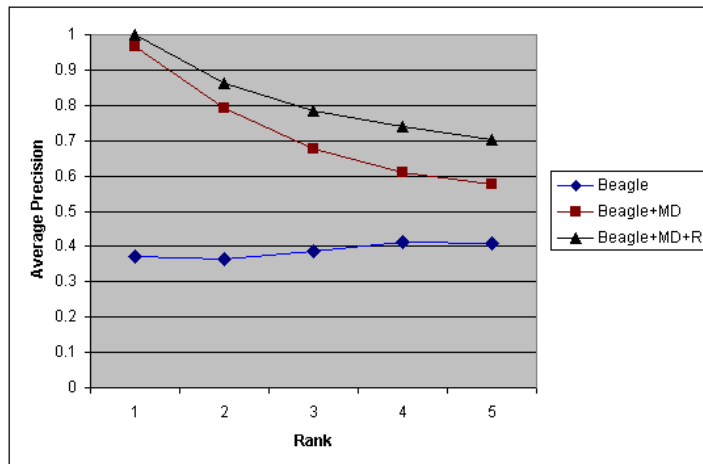


Fig. 6. Average Precision Results

used, as the datasets they cope with are too large. In this paper, we consider this amount to be equal to the total number of (unique) relevant results returned by the three systems we investigated. For every query, each system returned 5 results, 15 in total. Thus, the minimum possible total number of relevant results is 0 and the maximum is 15. Similarly, the maximum number of relevant results a system can return is 5 (since it only outputs 5 results), indicating that the recall will not necessarily be 1 when restricting the computation to rank 5. This version of recall is called *relative recall* [3].

5.3 Results and Discussion

As the main purpose of our experimental analysis was to produce a first estimate of each system's performance, we averaged the precision values at each rank from one to five for all 30 queries submitted by our experts. The results we obtained are depicted in Figure 6. We first notice that the current Beagle Desktop Search is rather poor, containing more qualitative results towards rank 4 to 5, rather than at the top of the result list. This is in fact explainable, since Beagle only uses TFxIDF to rank its results, thus missing any kind of global importance measure for the desktop resources. On the contrary, our first prototype, consisting of Beagle enhanced with RDF metadata annotations, already performs very well. An important reason for this high improvement is that metadata are mostly generated for those resources with high importance to the user, whereas the other automatically installed files (e.g., help files) are not associated with metadata, and thus ranked lower. When we have metadata describing a desktop item, more text is inherently available to search for, and thus this item is also easier to find. Finally, the precision values are even higher for our second prototype, which adds our desktop ranking algorithm to the metadata-extended version of Beagle. Clearly, ranking pushes our resources of interest more towards the top of the list, yielding even higher desktop search output quality.

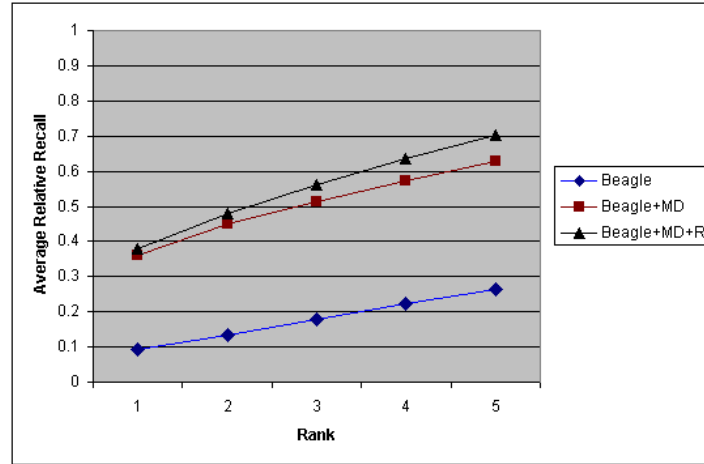


Fig. 7. Average Relative Recall Results

In the second part of the evaluation, we drew similar conclusions with respect to the average recall values (depicted in Figure 7): The recall of Beagle is very low, whereas that of our prototypes is almost three times better (owing to the additional information available as metadata, especially comments to the paper included in the emails). The difference between our prototypes is relatively small, which is correct, since recall analyzes the amount of good results returned, and both our systems yield relevant results. We thus conclude that enhancing Beagle with RDF metadata annotations significantly increases its recall (as metadata usually represents additional, highly relevant text associated to each desktop file), whereas adding desktop ranking further contributes with a visible improvement in terms of precision.

6 Conclusions and Future Work

We presented two main contributions that enhance traditional desktop search, focusing on how regular text-based desktop search can be enhanced with semantics / contextual information and ranking exploiting that information. Searching for resources then will not only retrieve explicit results but also items inferred from the users' existing network of resources and contextual information. Maintaining the provenance of information can help the search engine take into account the recommendations from other users and thus provide more retrieved results. The ranking module, by exploiting contextual information, improves retrieval precision and presentation of search results, providing more functionality to desktop search.

There are quite a few interesting additional contexts, that are worth investigating in the future: metadata embedded in multimedia files, the relations between objects embedded within each other (a presentation including pictures, tables, charts, etc.), or chat history. A further interesting question we want to

investigate in the future is how to learn contextual authority transfer weights from user feedback on ranked search results. Additionally we are experimenting with extended INEX datasets³ to evaluate the performance of our system on larger data sets.

7 Acknowledgements

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

References

1. B. Aleman-Meza, C. Halaschek, I. B. Arpinar, and A. Sheth. Context-aware semantic association ranking. In *Semantic Web and Databases Workshop*, 2003.
2. Apple spotlight search. <http://developer.apple.com/macosx/tiger/spotlight.html>.
3. R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.
4. A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB*, Toronto, Sept. 2004.
5. P. A. Chirita, R. Gavriloaie, S. Ghita, W. Nejdl, and R. Paiu. Activity based metadata for semantic desktop search. In *Proc. of the 2nd European Semantic Web Conference*, Heraklion, Greece, May 2005.
6. A. Damian, W. Nejdl, and R. Paiu. Peer-sensitive objectrank: Valuing contextual information in social networks. In *Proc. of the International Conference on Web Information Systems Engineering*, November 2005.
7. L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. C. Doshi, and J. Sachs. Swoogle: A search and metadata engine for the semantic web. In *Proc. of the 13th ACM Conf. on Information and Knowledge Management*, 2004.
8. S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff i've seen: A system for personal information retrieval and re-use. In *SIGIR*, 2003.
9. S. Ghita, W. Nejdl, and R. Paiu. Semantically rich recommendations in social networks for sharing, exchanging and ranking semantic context. In *Proc. of the 4th International Semantic Web Conference*, 2005.
10. Gnome beagle desktop search. <http://www.gnome.org/projects/beagle/>.
11. Google desktop search application. <http://desktop.google.com/>.
12. Google search engine. <http://www.google.com>.
13. T. Iofciu, C. Kohlschütter, W. Nejdl, and R. Paiu. Keywords and rdf fragments: Integrating metadata and full-text search in beagle++. In *Proc. of the Semantic Desktop Workshop held at the 4th International Semantic Web Conference*, 2005.
14. Msn desktop search application. <http://beta.toolbar.msn.com/>.
15. A. Sidiropoulos and Y. Manolopoulos. A new perspective to automatically rank scientific conferences using digital libraries. In *Information Processing and Management 41 (2005) 289qZ12*, 2005.
16. N. Stojanovic, R. Studer, and L. Stojanovic. An approach for the ranking of query results in the semantic web. In *ISWC*, 2003.
17. J. Teevan, C. Alvarado, M. Ackerman, and D. Karger. The perfect search engine is not enough: A study of orienteering behavior in directed search. In *Proc. of CHI*, 2004.

³ <http://inex.is.informatik.uni-duisburg.de/2005/>