Activity Based Metadata for Semantic Desktop Search

Paul Alexandru Chirita, Rita Gavriloaie, Stefania Ghita,
Wolfgang Nejdl, and Raluca Paiu

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

Abstract. With increasing storage capacities on current PCs, searching the World Wide Web has ironically become more efficient than searching one's own personal computer. The recently introduced desktop search engines are a first step towards coping with this problem, but not yet a satisfying solution. The reason for that is that desktop search is actually quite different from its web counterpart. Documents on the desktop are not linked to each other in a way comparable to the web, which means that result ranking is poor or even inexistent, because algorithms like PageRank cannot be used for desktop search. On the other hand, desktop search could potentially profit from a lot of implicit and explicit semantic information available in emails, folder hierarchies, browser cache contexts and others. This paper investigates how to extract and store these activity based context information explicitly as RDF metadata and how to use them, as well as additional background information and ontologies, to enhance desktop 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 nowadays, 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 due to powerful ranking algorithms like the PageRank algorithm introduced by Google [16]. The recent arrival of desktop search applications, which index all data on a PC, promises to increase search efficiency on the desktop. Still, these search applications are weaker than their web counterparts as they cannot rely on PageRank-like ranking mechanisms which have revolutionized web search. Unfortunately, they also fall short of utilizing desktop specific characteristics, especially context information. Some of these missed opportunities include:

- *Email context* is not utilized by the existing search algorithms, even though this clearly drops useful information. For example, one email might contain a question describing the object one is looking for, and another email in the same thread might include the answer to that question in the form of an attached document.

A. Gómez-Pérez and J. Euzenat (Eds.): ESWC 2005, LNCS 3532, pp. 439–454, 2005.

- Email attachments lose all contextual information as soon as they are stored on the PC, even though emails usually include additional information about their attachments, such as sender, subject, comments. We might discuss a paper with a colleague during a brainstorming session, and then afterwards send her the electronic version via email, together with a few helpful comments. After a while, our colleague might not remember details about the paper itself, but rather recall with whom she discussed it or which question was raised in the discussion and included as comment in the email. It would be helpful to find the stored paper not only based on its content, but also associatively based on that context[1].
- *Folder hierarchies* are barely utilized by the search algorithms, even though we might have spent considerable time to build sophisticated classification hierarchies for the documents we store. For example, pictures taken in Hanover are probably stored in a directory entitled ”Germany”, “Lower Saxony” or ”Hanover”, and it would be nice if we could utilize this information when we search for the pictures.
- *Browser caches* include all information about user’s browsing behaviour, which are useful both for finding relevant results (for example, if we remember how to find the project’s home page, but not the corresponding API specification), and for providing additional context for results. It would also be very useful if our search application not only returns one specific scientific paper we downloaded from the CiteSeer repository, but all the referenced and referring papers which we downloaded on that occasion as well.

As studies have shown that people tend to associate things to certain contexts [9], all this information should be utilized during search. So far, however, neither has this information been collected, nor have there been attempts to use it.

In this paper we discuss how to enhance and contextualize desktop search based on semantic metadata collected from different contexts available and activities performed on a personal computer. We explore three important contexts: electronic mail, folder hierarchies, and web cache. Analogously, other contexts might be exploited as well. We describe the semantics of these different contexts by appropriate ontologies and show how to extract and represent the corresponding context information as RDF metadata which can be used by a search application together with a full text index of our documents.

The next section gives an overview over existing approaches which try to exploit metadata in search algorithms, and classifies them according to how they use metadata to enhance search. Section 3 then shows how to describe contexts and their corresponding metadata by means of appropriate ontologies and association rules, and how to use these metadata in four different search scenarios where a simple full text index employed by current desktop search engines fails to find the information we are looking for. Finally, section 4 describes the architecture of our semantic desktop search environment, as well as our prototype.

[1] Desktop Search is in fact ”a search into our past”, and it should therefore exploit the associative functionality of the human memory.

2 Using Semantic Metadata in Search: A Classification

2.1 Using Metadata to Enrich Search Results

One of the most interesting semantic search efforts is probably being performed in the TAP project [8]. TAP builds upon the TAPache module, which provides a platform for publishing and consuming data from the Semantic Web. Its knowledge base is updated with the aid of the onTAP system, which includes 207 HTML page templates, being able to read and extract knowledge from 38 different high quality web sites. The key idea in TAP is that for specific searches, a lot of information is available in catalogs and backend databases, but not necessarily on Web pages crawled exhaustively by Google. The semantic search based results are independent of the results obtained via traditional information retrieval technologies and aim to augment them.

While searching for musicians and other well-known entities like cities, countries and others can draw upon the fact that a lot of information about them is available in backend databases, whose data sets can be joined based on the ID of that entity, the situation is different in the educational context, where topic classification is the most important characteristics of a page. This latter approach is used in our personal reader system [3], which finds additional pages related to the pages contained in a course, and again provides these as additional information to the core information presented.

2.2 Using Metadata to Connect and Visualize Information

In ”The Social Semantic Desktop” [2], the authors envision that the next step towards communication is a desktop application based on the Semantic Web, which could draw connections between all the types of data people interchange. For example, an entry in an agenda would be correlated with the author of an article or to the context associated to an email. Altogether, the entire information existing in a social network would be connected to each desktop. Such a structure would then help people organize and find information, due to the enhancement brought by metadata into the system.

The Fenfire project [5] proposes a solution to interlink any kind of information on one’s desktop. That might be the birthday with the person’s name and the articles she wrote, or any other kind of information. The idea is to make the translation from the current file structure to a structure that allows people to organize their data closer to the reality and to their needs, in which making comments and annotations would be possible for any file.

Haystack [17] pursues similar goals as Fenfire. One important focus is on working with the information itself, not with the programs it is usually associated with. For example, only *one* application should be enough to see both a document, and the email address of the person who wrote it. Therefore, a user could build her own links to Semantic Web objects (practically any data), which could then be viewed as thumbnails, web pages, taxonomies, etc.

A third project building an information management environment for the desktop is Gnowsis [19]. The main idea behind applications in this environment is the use of a central information server which allows users to administer and directly access all the information on their computer (for example the author of a file, her email address, etc.). Gnowsis envisions appropriate ontologies at four levels. The first one is used on the server, as it needs custom formats for the internal operation data and for its configuration files. The second one is for each application and the data stored by it. For example, in Outlook Express the types of data that can be found are emails, contacts and appointments. On the third level we have public ontologies, created by others to describe people, projects or documents (e.g. Dublin Core or FOAF). On the uppermost level, the user can create user-specific ontologies to fit her needs. For each level, only general architectural information is given, but no specific details or examples about the proposed ontologies, though.

In the context of another interesting prototype, the interface proposed by [21] improves image search by providing and using faceted metadata. Users can add flat or hierarchical categories of information to images, and then use them for filtering search results. Again, the idea is to provide an enhanced access to information, based on the different kinds of collected metadata.

2.3 Using Context Metadata to Find Information

[15] describes a very interesting approach for exploiting additional metadata for retrieving pictures. Their main idea is to rely on mostly automatically generated metadata (location, time and other digital photo metadata) and some manual annotations (events etc.) and to enhance these metadata automatically to provide information about actual light status (night, day, dawn, dusk), weather status and temperature, and additional aspects on the events, and then use these metadata to find stored images.

Another semantic search algorithm is proposed by [18]. It debuts with a classical text-based search on the metadata, whose output is then extended using the RDF network induced by the relations between semantic concepts, and finally reordered with techniques adapted from information retrieval.

[20] presents a new approach to content-based image retrieval. To improve the retrieval performance, the authors use a self-adjustable meta-database, which records the optimized relevance feedback information, representing the results obtained from previous queries from users that give a feedback on the relevance of the retrieved pictures. This kind of information partitions the images into classes denoting relevant images for future queries. The features taken into account by the algorithm are only low-level ones, though, such as HSV color-histograms or directional histograms.

3 Integrating Context Metadata Within Desktop Search

3.1 How Do Users Search?

Now how can we enhance desktop search with additional metadata? Clearly, if we know how users search, we can support their queries in an appropriate way. Recent studies of user web search behavior [4] have shown that the user goals can be classified into three main categories:

- *Navigational*: the user is searching for a specific web site, whose URL she forgot.
- *Informational*: the user is looking for information about a topic she is interested in.
- *Resource Seeking*: the user wants to find a specific resource (e.g. lyrics of a song, a program to download, a map service, etc.).

On our computer we are mainly interested in navigational queries, i.e. the user knows she stored a resource somewhere on the PC and now wants to find it again. Other less frequent, but possible, search goals are resource seeking (for example when searching for a previously installed application which plays MPEG-4 movies) and the close-directed subclass of informational queries [4] (searching for a resource annotated with a given description, such as ”introduction to logic programming”). The other types of informational queries are almost inexistent on the desktop, as one generally has at least a vague picture of what is stored, and thus knows whether resources on a specific topic do exist on the PC or not[2].

Now clearly, when searching for something on our desktop we want to be able to exploit as much additional context as possible. In the following sections, we will discuss which context information is available for desktop search, how we can describe this context information using appropriate ontologies and how we can represent this information by explicit or inferred RDF metadata.[3]

After a brief presentation of current conventional approaches to desktop search (Section 3.2), we will analyze three important contexts which can be exploited to enhance desktop search: emails in Section 3.3, directory structures in 3.4, and the web cache in 3.5 and 3.6. For each context, we describe ontologies representing the available context information, and discuss both explicitly available metadata, as well as metadata that can be inferred and materialized using appropriate association rules.

3.2 Current Approaches to Desktop Search

The difficulty of accessing information on our computers has prompted several first releases of desktop search applications during the last months. The most prominent examples include Google desktop search [7] (proprietary, for Windows) and the Beagle open source project for Linux [6]. 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* [1]) 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, last modification date, or metadata embedded into specific files (images taken with digital cameras for example include many additional characteristics, such as exposure information or whether a flash was used). While this is indeed an improvement over regular search, it still misses contextual information often

[2] If she knows that ”something” is there, then the search becomes “navigational” or ”resource seeking”. If she knows there is nothing stored on the given topic, she would not search for it on her desktop.

[3] Note, that even inferred metadata have to be materialized in order to enable efficient search.

resulting or inferable from explicit user actions or additional background knowledge, as discussed in the next sections.

In the following we will introduce four important search contexts, each with a small scenario, where ordinary full-text search fails, but additional context metadata provide the necessary information for finding the document we search for. For each context we will describe RDFS ontologies defining the metadata relevant for that context, as well as association rules and possible background knowledge which infer and materialize additional metadata.

3.3 Exploiting E-Mail Context

Scenario. Alice is interested in distributed page ranking, as her advisor asked her to write a report to summarize the state of the art in this research area. She remembers that during the last month she has discussed with a colleague about a distributed PageRank algorithm, and also that the colleague sent her the article via email. Though the article does not mention distributed PageRank, but instead talks about distributed trust networks, it is basically equivalent to distributed PageRank as her colleague remarked in this email. Obviously she should be able to find the article based on this additional information.

Context and Metadata. There are several aspects relevant to our email context. Sender and receiver fields of the email are clearly relevant pieces of information. Further information can be captured if we analyze the date of the email or the "*reply_ to*" field, which gives thread information and is useful to determine social network information in general, for example which people discussed which topic etc.

Metadata should be generated automatically while the user works. For example, when an email is received, the system automatically generates email RDF metadata, instantiating e.g. "*To*", "*From*" and "*Comment*" metadata from the email fields, and associating them to the document(s) attached to this email.

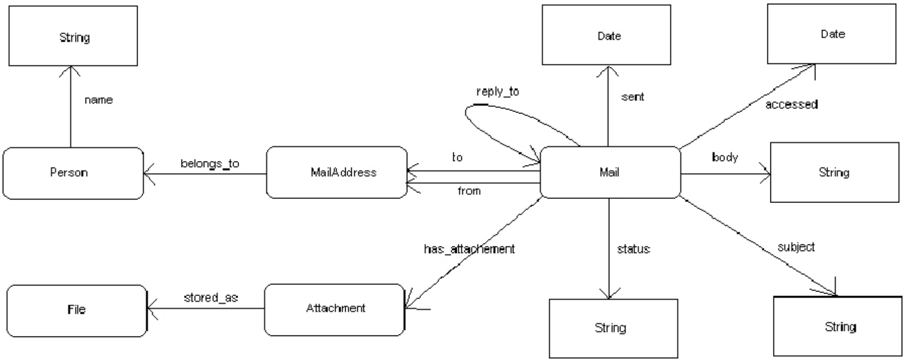

Fig. 1. Email prototype

Useful RDFS Ontologies. Basic properties for this context are properties referring to the date when an email was sent or the date it was accessed, the subject of the email and the email body. The status of an email can be described as seen/unseen or read/unread. We also have a property of the type *reply_to* which represents thread information. The *has_attachment* property describes a 1:n relation because a mail can have one or more attachments. The *to* and *from* properties connect to Class *MailAddress* which connects to Class *Person*. A *Person* is usually associated to more than one *MailAddress* instances. For attachments we keep the connection to the email it was saved from, because when we search for an attachment we want to use all attributes originally connected to the email it was attached to. The *stored_as* attribute is the inverse relation of the *File:stored_from* property we will see later.

Corresponding Association Rules. Association rules infer and materialize additional metadata information. For example, when creating the annotations, for each stored file we also associate a subject, derived from the subject of the email the file was attached to. The corresponding association rule, written in Datalog style, looks as follows:

$$\begin{aligned}subject(File, Subject) \leftarrow\ & stored\_as(Attachment, File),\\ & has\_attachment(Mail, Attachment),\\ & subject(Mail, Subject).\end{aligned}$$

Similarly, we also associate date and body text to the attached documents:

$$\begin{aligned}accessed(File, Date) \leftarrow\ & stored\_as(Attachment, File),\\ & has\_attachment(Mail, Attachment), accessed(Mail, Date).\\ body(File, Body) \leftarrow\ & stored\_as(Attachment, File),\\ & has\_attachment(Mail, Attachment), body(Mail, Body).\end{aligned}$$

as well as the name of the sender of the original email:

$$\begin{aligned}from(File, Name) \leftarrow\ & stored\_as(Attachment, File),\\ & has\_attachment(Mail, Attachment),\\ & from(Mail, MailAddress),\\ & belongs\_to(MailAddress, Person), name(Person, Name)\end{aligned}$$

In email threads connected through the *reply_to* relationship, we also inherit email subjects and bodies in addition to the original email subject / body:

$$\begin{aligned}& subject(Mail, Subject).\\ & subject(Mail, Subject) \leftarrow reply\_to(Mail, Mail_1), subject(Mail_1, Subject).\\ & body(Mail, Body).\\ & body(Mail, Body) \leftarrow reply\_to(Mail, Mail_1), body(Mail_1, Body).\end{aligned}$$

Note that these association rules generate and materialize the appropriate metadata before the query is evaluated, and thus materialized metadata can be used directly during search, similar to the full text of the file / document. In our example, we can retrieve the correct document by using body text and sender information associated to this document, inherited from the original email.

3.4 Exploiting File Hierarchy Context

Scenario. In our second scenario, Alex spent his holiday in Hanover, Germany, taking a lot of digital pictures. He usually saves his pictures from a trip into a folder named after the city or the region he visits. However, he has no time to rename each image, and thus their file names are the ones used by his camera (for example ”DSC00728.JPG”). When he forgets the directory name, no ordinary search can retrieve his pictures, as the only word he remembers, ”Germany”, does neither appear in the file names, nor in the directory structure. It would certainly be useful if an enhanced desktop search with “pictures germany” would retrieve his Hanover pictures.

Context and Metadata. In this example we need to consider file type and directory name information, and we need to be able to go beyond simple keyword search, taking part-of relationships and synonyms into account. To enrich the context metadata provided by file and directory names, we use WordNet [13], a lexical reference system which contains English nouns, verbs, adjectives and adverbs organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets. In our case, we use the following additional relationships:

- *Hypernym*: Designates a class of specific instances. X is a hypernym of Y if Y is a (kind of) X.
- *Holonym*: Designates the superset of an object. A is a holonym of B if B is a part of A.
- *Synonyms*: A set of words that are interchangeable in some context. X is a synonym of Y if Y can substitute X in a certain context without altering the meaning.

Useful RDFS Ontologies. Obviously, our context metadata for files include the basic file properties like date of access and creation, as well as the file owner. File types can

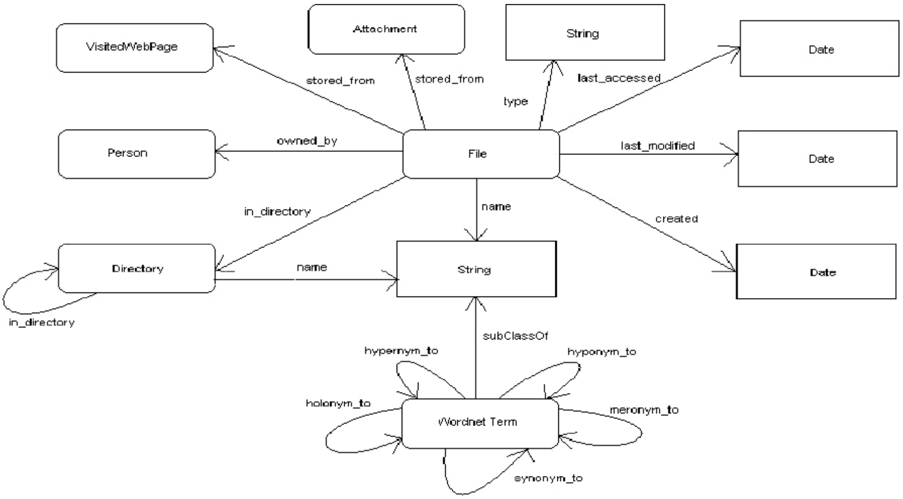

Fig. 2. File prototype

be inferred automatically, and provide useful information as well (in our case, the file is of type "JPEG image data"). Additionally, a file might be a visited web page which we stored on our computer or an attachment saved from an email. This *stored_from* property is of great importance because this represents information that current file systems miss, the provenance of information. We also keep track of the whole file path, including the directory structure. Finally, we extend the strings used in name and type metadata using WordNet information: synonyms, hypernyms, and holonyms. For each term we add the information provided by WordNet in order to enrich the context of the stored file.

Corresponding Association Rules. The use of WordNet induces the following association rules:

$$name(File, String_1) \leftarrow name(File, String_2), synonym\_to(String_2, String_1).$$
$$name(File, String_1) \leftarrow name(File, String_2), holonym\_to(String_2, String_1).$$
$$name(File, String_1) \leftarrow name(File, String_2), hypernym\_to(String_2, String_1).$$

Furthermore, we associate directory names as additional names to the contained files as well. The rules allow us to add explicit part-of information ("Hanover is part of Germany"), as well as synonym information ("picture" is a synonym to "image"), and enable us to successfully solve the search problem discussed in our scenario.

3.5 Exploiting the Web Cache Context for Visualization

Scenario. Even though Web search engines are providing surprisingly good results, they still need to be improved to take user context and user actions into account. Consider for example Paul, who is looking for the Microsoft internships web page, which he has previously visited, coming from the Microsoft main home page. If he does not remember the right set of keywords to directly jump to this page, it certainly would be nice if enhanced desktop search, based on his previous surfing behavior, would support him by returning the Microsoft home page, as well as providing the list of links from this page he clicked on during his last visit.

Context and Metadata. The context we have to use here can be extracted from Paul's web cache, so we want to annotate each cached web page with additional information both for its basic properties (URL, access date, etc.), as well as more complex ones such as the used in-going and out-going links to other neighboring pages, reflecting Paul's surfing behavior. This way, when browsing a certain cached page, enhanced desktop search can also provide information about the context in which that document has been useful for the user, i.e. how it was reached or which links were followed from there.

Useful RDFS Ontologies. Correspondingly, the central class in this scenario's ontology is the class *VisitedWebPage*. Upon visiting a web page, the user is more interested in the links she has used on that page, rather than every possible link which can be followed from there. Thus, the metadata contains only the hyperlinks *accessed* for each stored web page:

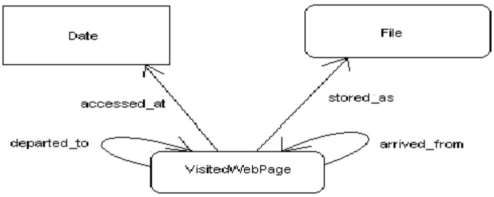

Fig. 3. WebPage prototype

- *departed_to* is a relation of the type one to many (as the user could have accessed many pages from a web page) which shows the hyperlinks the user clicked on the current web page;
- *arrived_from* is a relation representing the page(s) the user came from.

Also here, we have added properties related to the time of access and place of storage in the hard disk cache. For specific scenarios we can define subclasses of this base class, which include scenario specific attributes, for example recording the browsing behavior in CiteSeer, which we will discuss in the next section.

Corresponding Association Rules. There are no specific association rules materializing inferred metadata we need for our scenario. Instead we use our metadata for enriching search results. Displaying context information for enhanced browsing under this scenario uses a similar layout as the TAP search screen, with the web pages or documents from the cache provided in the main window, and an additional frame to display the context information using the *departed_to* and *arrived_from* relations.

3.6 Exploiting the Web Cache Context to Enrich Search Results

Scenario. If we have more information about the web pages visited, we can provide even better context information. Suppose that Alice browses through CiteSeer for papers on a specific topic, following reference links to and from appropriate papers, and downloads the most important documents onto her computer. Now as soon as they are stored in one of her directories, her carefully selected documents are just another bunch of files without any relationships. They have completely lost all information present in CiteSeer, in this case which paper references specific other papers or is referenced by another paper, and which papers Alice deemed important enough not only to look at but also to download. It is the task of a semantic desktop search environment to preserve that information and make it available as explicit metadata.

Context and Metadata. As discussed, stored files on today's computers do not tell us whether they were saved from a web page or from an email, not to mention the URL of the web page, out-going or in-going visited links and more specific information inferable from this information and a model of the web page context browsed, as discussed in our scenario. All this information should be covered by our metadata to connect the stored files to their original contexts, and thus allow the user to exploit all the previous knowledge and context she gathered around them.

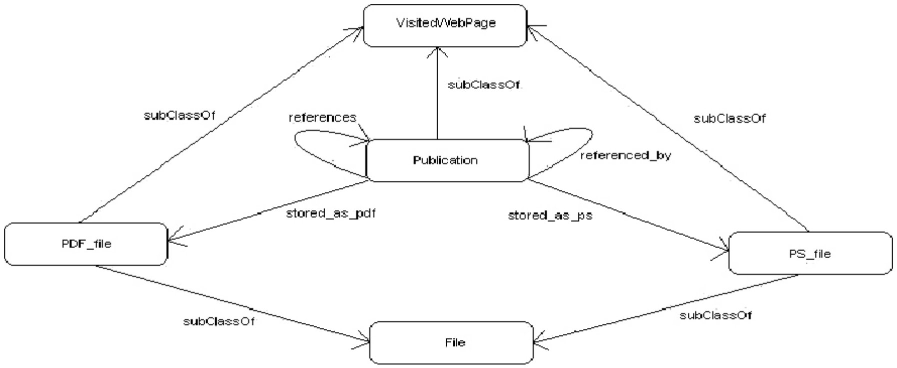

Fig. 4. Publication prototype

Useful RDFS Ontologies. In our scenario we make use of additional knowledge about how CiteSeer pages are connected. We therefore create a subclass of *VisitedWebPage* called *Publication*, and add suitable properties as described in figure 4. The *Publication* class represents a CiteSeer document web page. It records the CiteSeer traversed links from that page using the *references* property and the CiteSeer documents which the user visited before using the *referenced_by* property. It is easy to notice that these pages represent a subset of the metadata captured by the *departed_to* and *arrived_from* relations. *PDF_file* and *PS_file* are subclasses of *File*, and are connected to *Publication* with subproperties of "*stored_as*", namely "*stored_as_pdf*" and "*stored_as_ps*".

Corresponding Association Rules. In our semantic desktop search environment we use these metadata to enrich the search results by displaying the context of the document found in the form of downloaded papers referencing that document, or downloaded papers referenced by the document. This can be expressed for example by an association rule such as the following one:

$$\begin{aligned} downloaded\_references(Document, File) \leftarrow \\ stored\_as(Publication_1, Document), \\ references(Publication_1, Publication_2), \\ stored\_as(Publication_2, File). \end{aligned}$$

4 Desktop Search Architecture and Prototype

4.1 Generating Input Metadata

Event Triggered Metadata Generation. The main characteristic of our 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 for example 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 we use in our prototype.

Metadata Generator Applications. Depending on the type and context of the file / event, metadata generation is then performed by appropriate metadata generator applications, as described in the next paragraphs. These applications build upon an appropriate RDFS ontology as described in the previous sections 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 the appropriate association rules plus possibly some additional background knowledge (e.g. the WordNet ontology in our prototype). 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.

The architecture of our prototype environment is depicted in Figure 5. It includes three prototype metadata generator applications, based on the scenarios described in the previous section 3. We will shortly describe them in the following paragraphs.

Email Metadata Generator. Our current email prototype is built on top of the JavaMail API [10]. It processes the incoming emails into a separate class, derived from the *Message* class defined in JavaMail. The associated metadata is easily generated according to figure 1, as the *Message* class already provided helpful methods in this direction (e.g. "*getTo*", "*getRecipients*", "*getSubject*" and "*getSentDate*"). Further metadata are generated when attachments are stored in the file system. Metadata are stored as RDF using the Jena toolkit [11]. Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF and RDFS, including a rule-based inference engine which we use to implement our association rules.

File Metadata Generator. Upon creation of a new file, its path is decomposed into a sequence of tokens, one for each level of the directory tree existing on the hard disk. Each of these tokens is added as metadata description to the file, together with the usual file attributes, as described in section 3.4. We use WordNet to add additional metadata (WordNet senses) both to the file name and to each token of the path, thus capturing all meaningful information implicitly available through the file and folder names. Our file prototype is again implemented in Java, and uses the JWNL API [12] to access the WordNet relational dictionary. As in the previous module, we also use the Jena API to generate the RDF file that contains the annotation corresponding to the file structure, in which each indexed file is a resource.

As future work, we intend to extract additional specific information stored in several widely used file types. For example, many image formats provide specific additional metadata, such as exposure information. Another possible improvement for this generator is to use additional background knowledge about seasons etc., as well as to let the user manually add more annotations to files or directories. We could then search for the pictures we took during the last winter in Germany, or during a special event in our life, like a birthday.

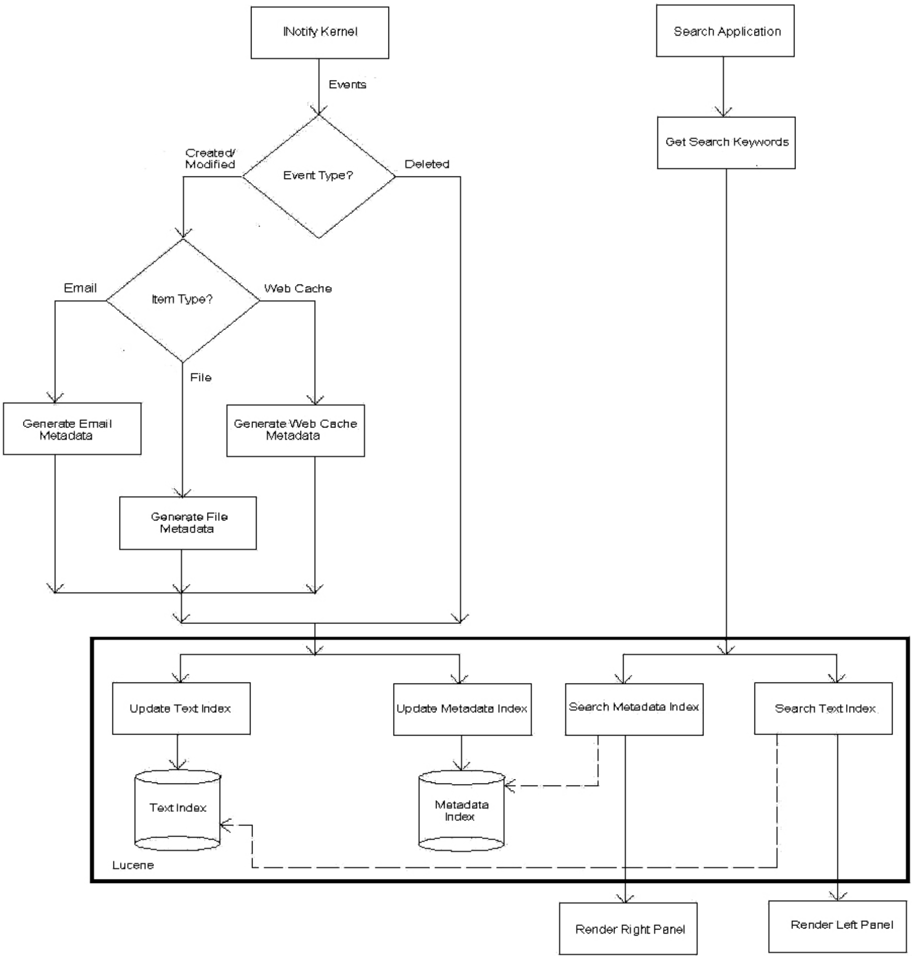

Fig. 5. Prototype Application Architecture

Web Cache Metadata Generator. In the web cache prototype, the annotation of the cached web pages is triggered by browsing web pages which were not previously stored in the local cache. Generation starts with the basic annotations for each web page (e.g. access date) and then proceeds with the annotations representing the connections between web pages (for example from which page did the user arrive at the current one, or which hyperlinks of the current page are traversed). Again, we use the Jena API to export the annotations in RDF format.

For specific sites, the metadata generator uses additional ontologies. In our prototype this is done when using the CiteSeer repository. These ontologies then trigger additional metadata generation, which can be used during search, as well as for enriching search results.

4.2 Displaying and Enriching Search Results

Enhanced semantic desktop search provides a search service similar to its web sibling. However, rather than searching only one through the full-text index, it also searches the additional metadata index, with each metadata item linked to the resource it has been derived from.

The regular search interface is as simple as the one provided by Google, i.e. an input text box for the searched terms and a search button. This type of search looks for the keywords in both indexes automatically. Results are then presented as in TAP [8]: the left side of the output window displays the hits matched from the text index, and the right side contains additional information provided through the metadata associated with the chosen result document.

The items displayed on the right hand side obviously depend on the type of the result document. For the web cache scenario this allows us to show not only the previously browsed pages, but also the entire context in which they have been used and accessed (for example where did the user go from each page, or which referenced papers the user downloaded related to a found document). This helps a lot, as the user now has all the orienteering steps [9] right in front of her.

An additional advanced search interface allows the user to restrict her search to one of the two indexes. Moreover, she can define filters, by choosing where to search (only in the emails, files or the web cache), each category also allowing other additional filters according to its ontology (e.g. *To*, *From*, *Reply To*, *Subject*, *Attachment*, etc. in the email scenario).

5 Conclusions and Further Work

Advanced desktop search needs semantics and metadata. Applying search engine technology on the desktop is useful, but not sufficient, because sophisticated heuristics and algorithms like PageRank, which are very successful on the web, do not work on the desktop. On the other hand, our personal desktop environment provides a lot of context not available on the web, which can be used to implement sophisticated semantic search functionalities on our desktop surpassing those possible on the web.

This paper presents concept, architecture and prototype for a semantic desktop search environment, which promises to exploit the information present in these contexts, accumulated by user activities and additional background knowledge. Our search environment relies on ontologies describing appropriate metadata for different contexts relevant on the desktop and uses these semantic annotations to both extend search functionalities and enrich search results.

The semantic desktop search environment contains two distinct modules. The first one is *the index*, which consists of a metadata repository including all metadata associated to each resource on the desktop, as specified in the appropriate context ontologies, and a regular search engine full-text index of these resources. The second module is *the search module*, which combines keyword search on the full-text index with semantic search on the metadata repository to provide both improved functionalities for finding information on our PC, as well as enriching the search results and visualizing existing contexts using the additional knowledge stored in the metadata repository.

Comparing the possibilities for a semantic desktop search environment to semantic search on the web, we believe that semantic web technologies might ultimately be more important on the desktop than on the web. This is because, first, our desktop environment is “limited” in the sense that we will be able to describe most relevant contexts rather easily, and thus will be able to provide more complete ontologies / metadata specifications for the desktop environment than for the web in general. Second, even with 200GB hard disks in our computers, the amount of data and metadata itself is limited compared to the information available on the web, so more sophisticated algorithms for using semantic annotations are feasible on the desktop than on the web.

We are currently working on integrating our metadata repository and tools into one of the existing approaches to desktop search, Gnome Beagle [6], where we can re-use their conventional infrastructure for full-text search on the desktop. Additionally, we are extending our available context ontologies and metadata generation functionalities beyond the current status as described in this paper, in conjunction with several user surveys meant to capture both the requirements of a larger set of users, as well as to measure the improvements provided by adding semantic annotations to desktop search.

References

1. Apple spotlight search. http://developer.apple.com/macosx/tiger/spotlight.html.
2. Stefan Decker and Martin Frank. The social semantic desktop. In *DERI Technical Report 2004-05-02*, 2004.
3. P. Dolog, N. Henze, W. Nejdl, and M. Sintek. Personalization in distributed elearning environments. In *Proceedings of the 13th World Wide Web Conference*, 2004.
4. Rose D. E. and Levinson D. Understanding user goals in web search. In *Proc. of WWW 2004, May 17-22, 2004, New York, USA*, 2004.
5. Benja Fallenstein. Fentwine: A navigational rdf browser and editor. In *Proceedings of 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, 2004.
6. Gnome beagle desktop search. http://www.gnome.org/projects/beagle/.
7. Google desktop search application. http://desktop.google.com/.
8. R. Guha, Rob McCool, and Eric Miller. Semantic search. In *Proceedings of the twelfth international conference on World Wide Web*, pages 700–709. ACM Press, 2003.
9. Teevan J., Alvarado C., Ackerman M. S., and Karger D. R. The perfect search engine is not enough: A study of orienteering behavior in directed search. In *In Proc. of CHI*, 2004.
10. Javamail api. http://java.sun.com/products/javamail/.
11. Jena api. http://jena.sourceforge.net/.
12. Jwnl api. http://sourceforge.net/projects/jwordnet.
13. G.A. Millet. Wordnet: An electronic lexical database. *Communications of the ACM*, 38(11):39–41, 1995.
14. Msn desktop search application. http://beta.toolbar.msn.com/.
15. Mor Naaman, Susumu Harada, Qian Ying Wang, Hector Garcia-Molina, and Andreas Paepcke. Context data in geo-referenced digital photo collections. In *Proceedings of the 12th annual ACM International Conference on Multimedia*, 2004.
16. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.
17. Dennis Quan and David Karger. How to make a semantic web browser. In *Proceedings of the 13th International WWW Conference*, 2004.

18. Cristiano Rocha, Daniel Schwabe, and Marcus Poggi de Aragao. A hybrid approach for searching in the semantic web. In *Proceedings of the 13th International World Wide Web Conference*, 2004.
19. Leopold Sauermann. Using semantic web technologies to build a semantic desktop. Master's thesis, TU Vienna, 2003.
20. Yimin Wu and Aidong Zhang. Category-based search using metadatabase in image retrieval. In *IEEE International Conference on Multimedia and Expo*, 2002.
21. Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *Proceedings of the conference on Human factors in computing systems*, 2003.