EXPLOITING METADATA FOR CONTEXT CREATION AND RANKING ON THE DESKTOP

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades

Doktorin der Naturwissenschaften

Dr. rer. nat.

genehmigte Dissertation von

Dipl.-Ing. Stefania Costache

geboren am 27. Dezember 1980, in Buzau, Rumänien

2010

Referent: Prof. Dr. Wolfgang Nejdl
Ko-Referent: Prof. Dr. Heribert Vollmer
Tag der Promotion: 1. Dezember 2010

ABSTRACT

With the ever increasing size of the number of resources we store on our computers, there is an obvious need for better tools for managing our personal information. First, there is a need of keeping resources connected beyond the simple folder hierarchies, in order to reflect the user working contexts and tasks. The main problem is that as soon as we store something on our computers, for example a file, then the connection to the email that it was sent with is immediately lost, and also the whole context around it. And second, while faced with this vast amount of data, even if we are able to search and exploit these connections, an ordering is very much needed. We need this not only for retrieving our own data from our PCs, but we also need to be able to give more importance to some resources coming from more trusted persons. In this thesis we propose several solutions not only for enhancing the current data with semantic connections in order to create contexts, but also for ranking resources both on the desktop, and in a collaborative environment where users exchange resources and need to take trust and privacy into account. Several experiments support the ideas proposed and also detail on various situations on which method is best to be applied.

We first focus on the enhancement of resources with context metadata, by recreating lost connections among them. We propose several modules fully integrated within the Beagle^{++} system and also show via experiments that these metadata generators are useful in finding resources. Also, time connections are exploited and we show that such connections are valuable, since they simulate the normal user behaviour when working on a task - several resources are accessed in a sequence rather frequently. Finally, we show how annotations can be a step further for extending the desktop to the Web - we automatically extract personalized annotations from within the desktop documents and use them for the annotation of visited web pages.

Then, we concentrate upon the benefits that a ranking mechanism can bring. We build on top of the PageRank algorithm a semantic ranking mechanism applied to the desktop, which fully exploits the time connections previously created. We also extend to the collaborative environment and show how recommendations coming from within the user's working group can be ranked, by taking into account the trust that he has in the persons that sent him those resources. Also, more trust and privacy issues are further explored on how we can share our resources but not disclosing the structure of our resources. A world node solution is proposed and we prove that it is a good trade-off between quality and privacy, given also various amounts of data that are fully shared between users.

Keywords: *Desktop Search, Ranking, Metadata Generation*

ZUSAMMENFASSUNG

Mit der zunehmenden Größe und Anzahl der Ressourcen, die wir auf unseren Computern speichern, gibt es eine offensichtliche Notwendigkeit für bessere Werkzeuge zur Verwaltung von unseren persönlichen Informationen. Erstens besteht der Bedarf an Verknüpfungen zwischen Ressourcen, über die einfachen Ordner-Hierarchien hinaus, den Arbeitskontext und Zusammenhänge von Aufgaben wiederzugeben. Das Hauptproblem ist, dass, sobald wir etwas auf unseren Rechnern speichern - zum Beispiel eine Datei aus dem Anhang einer E-Mail - geht die Verbindung zum gesamten Kontext woher die Datei stammt - z.B. die dazugehörige E-Mail - sofort verloren. Wir benötigen ein gutes Ranking, damit wir auch in der Lage sind - bei der Konfrontation mit großen Datenmengen - vorhandene Verbindungen bei der Suche zu nutzen. Wir brauchen dieses nicht nur für das Abrufen von Daten aus unseren eigenen PCs, wir müssen auch in der Lage sein, mehr Wert auf Ressourcen von vertrauenswürdigen Personen zu legen. In dieser Arbeit schlagen wir mehrere Lösungen vor, nicht nur für die Erweiterung der aktuellen Daten um semantische Zusammenhänge, sondern auch für das Ranking von Ressourcen auf dem Desktop sowie in einer kollaborativen Umgebung, in der Benutzer Ressourcen auswechseln und wo Vertrauen und die Privatsphäre berücksichtigt werden müssen. Mehrere Versuche unterstützen die vorgeschlagenen Ideen und geben an, welche Methoden für welche Situationen am besten angewandt werden sollen.

Wir konzentrieren uns zunächst durch Wiederherstellung verlorener Verbindungen auf die Erweiterung der Ressourcen um Kontext-Metadaten. Wir schlagen mehrere Module vor - vollständig in Beagle^{++} integriert - und zeigen mittels Experimenten, dass diese Metadaten-Generatoren nützlich bei der Suche nach Ressourcen sind. Außerdem werden Zeit-Verbindungen genutzt und wir zeigen, dass solche Verbindungen wertvoll sind, da sie das normale Nutzerverhalten bei der Arbeit an einer Aufgabe simulieren; mehrere Ressourcen werden ziemlich häufig in einer Sequenz aufgerufen. Schließlich zeigen wir, wie Anmerkungen die Erweiterung des Desktops auf das Web noch einen Schritt weiter bringen können; wir extrahieren personalisierte Anmerkungen aus Desktop-Dokumenten automatisch und nutzen diese für Annotationen von besuchten Webseiten.

Danach konzentrieren wir uns auf die Vorteile, die ein Ranking-Mechanismus bringen kann. Wir bauen auf den PageRank-Algorithmus auf und wenden einen semantischen Ranking-Mechanismus auf dem Desktop an, der die zuvor erstellten Verbindungen im vollen Umfang nutzt. Wir erweitern auch die kollaborative Benutzerumgebung und zeigen unter Berücksichtigung des Vertrauens in die Personen, die diese Ressourcen gesandt haben, wie die Empfehlungen innerhalb der Benutzer-Arbeitsgruppe gerankt werden können. Weitere Fragen zu Vertrauen und Privatsphäre werden erkundet, z.B. wie wir unsere Ressourcen offenlegen, aber nicht die Struktur dieser Ressourcen. Unsere vorgeschlagene Lösung ist ein guter Kompromiss zwischen Qualität und Privatsphäre, da verschiedenartige große Mengen von Daten vollständig zwischen den Nutzern geteilt werden.

Schlagwörter: *Desktop Search, Ranking, Metadata Generation*

FOREWORD

The work presented in this thesis has been published at various conferences, as follows.

In Chapter 2 we describe contributions included in:

- *Leveraging Personal Metadata for Desktop Search: The Beagle^{++} System.* Enrico Minack, Raluca Paiu, Stefania Costache, Gianluca Demartini, Julien Gaugaz, Ekaterini Ioannou, Paul-Alexandru Chirita, Wolfgang Nejdl. In: Journal of Web Semantics, 2010. [MPC+10]

- *Desktop Context Detection Using Implicit Feedback.* Paul-Alexandru Chirita, Stefania Costache, Julien Gaugaz, Wolfgang Nejdl. In: Proceedings of the Personal Information Management Workshop at the 29th Annual ACM International Conference on Special Interest Group on Information Retrieval. SIGIR'06, Seattle, WA, USA, August 6-11, 2006. [CCGN06]

- *Detecting Contexts on the Desktop Using Bayesian Networks.* Stefania Costache, Julien Gaugaz, Ekaterini Ioannou, Wolfgang Nejdl. In: Proceedings of the Desktop Search Workshop: Understanding, Supporting, and Evaluating Personal Data Search at the 33rd Annual ACM International Conference on Special Interest Group on Information Retrieval. SIGIR'10, Geneva, Switzerland, July 19-23, 2010. [CGIN10]

- *P-TAG: Large Scale Automatic Generation of Personalized Annotation TAGs for the Web.* Paul-Alexandru Chirita, Stefania Costache, Siegfried Handschuh, Wolfgang Nejdl. In: Proceedings of the 16th International World Wide Web Conference. WWW'07, Banff, Alberta, Canada, May 8-12, 2007. [CCNH07]

Chapter 3 presenting methods for computing ranking on the desktop and in a cooperative environment is built upon the work published in:

- *Activity Based Links as a Ranking Factor in Semantic Desktop Search.* Julien Gaugaz, Stefania Costache, Paul-Alexandru Chirita, Claudiu S. Firan, Wolfgang Nejdl. In: Proceedings of the 6th Latin American Web Congress. LA-WEB '08, October 28 - 30 2008, Vila Velha, Espirito Santo, Brasil. [GCC+08]

- *Semantically Rich Recommendations in Social Networks for Sharing, Exchanging and Ranking Semantic Context.* Stefania Ghita, Wolfgang Nejdl, Raluca Paiu. In Proceedings of the 4th International Semantic Web Conference. ISWC '05, Galway, Ireland, 6-10 November 2005. [GNP05a]

- *Personalizing PageRank-Based Ranking over Distributed Collections.* Stefania Costache, Wolfgang Nejdl, Raluca Paiu. In Proceedings of the 19th International Conference on Advanced Information Systems Engineering. CAiSE '07, Trondheim, Norway, 11-15 June 2007. [CNP07]

During my Ph.D. studies I have also published a number of papers investigating the use of metadata for improving desktop search, but also on how we can detect events from content generated by users, also known as social media, and more specific from blogs. This aspect is not touched in this thesis due to space limitation, but the complete list of publications follows:

- *The Beagle++ Toolbox: Towards an Extendable Desktop Search Architecture.* Ingo Brunkhorst, Paul A. Chirita, Stefania Costache, Julien Gaugaz, Ekaterini Ioannou, Tereza Iofciu, Enrico Minack, Wolfgang Nejdl, Raluca Paiu. In: Proceedings of the Semantic Desktop and Social Semantic Collaboration Workshop at the International Semantic Web Conference, ISWC '06, November 2006, Athens, GA, USA. [BCC+06]

- *Beagle++: Semantically Enhanced Searching and Ranking on the Desktop.* Paul A. Chirita, Stefania Costache, Wolfgang Nejdl, Raluca Paiu. In: Proceedings of the 3rd European Semantic Web Conference. ESWC '06, June 2006, Budva, Montenegro. [CGNP06]

- *Semantically Enhanced Searching and Ranking on the Desktop.* Paul A. Chirita, Stefania Costache, Wolfgang Nejdl, Raluca Paiu. In Proceedings of the International Semantic Web Conference Workshop on the Semantic Desktop - Next Generation Personal Information Management and Collaboration Infrastructure. ISWC '05, Galway, Ireland, November 2005. [CGNP05]

- *Semantically Rich Recommendations in Social Networks for Sharing and Exchanging Semantic Context.* Stefania Costache, Wolfgang Nejdl, Raluca Paiu. In: Proceedings of the 2nd European Semantic Web Conference Workshop on Ontologies in P2P Communities, ESWC '05, Greece, May 2005. [GNP05b]

- *Using Your Desktop as Personal Digital Library.* Stefania Ghita. In: Proceedings of the Doctoral Consortium at the 9th European Conference on Research and Advanced Technology for Digital Libraries, ECDL '05, Vienna, Austria, 18-23 September 2005. [Ghi05]

- *Task Specific Semantic Views: Extracting and Integrating Contextual Metadata from the Web.* Stefania Ghita, Nicola Henze, Wolfgang Nejdl, Raluca Paiu. In: Proceedings of the Workshop on The Semantic Desktop - Next Generation Personal Information Management and Collaboration Infrastructure at the 4th International Semantic Web Conference, ISWC'05, Galway, Ireland, November 2005. [MGHN05]

- *Application Independent Metadata Generation.* Jürgen Belizki, Stefania Costache, Wolfgang Nejdl. In: Proceedings of the International ACM Workshop on Contextualized Attention Metadata: Collecting, Managing and Exploiting of Rich Usage Information at the 15th ACM CIKM (Conference on Information and Knowledge Management), CAMA '06, Arlington, VA, USA, November 2006. [BCN06]

- *Query Ranking in Information Integration.* Rodolfo Stecher, Stefania Costache, Claudia Niederée, Wolfgang Nejdl. In: Proceedings of the 22nd International Conference on Advanced Information Systems Engineering, CAiSE'10, Hammamet, Tunisia, June 2010. [SCNN10]

Contents

List of Figures

1 Introduction

Nowadays, our personal information is mostly in electronic form, spanned over several physical locations, as desktop computers, PDAs, mobile phones, digital cameras, etc. The capacity of our hard drives today, combined with more and more powerful computers and explosion of communications in electronic form – be they emails or documents downloaded from the web – allows us to accumulate on our desktop an overwhelming quantity of personal information coming from several locations. All this information is useless unless we are able to find it at the time we *need* it.

1.1 Personal Information Management

The Personal Information Management (PIM) field is rather new, and aims at exactly solving these problems of offering to the users solutions for managing this huge amount of information. The ideal would be not to only have a tool that can give a nice overview of the resources, but a good use of them towards our every-day goals, which ultimately translate into a better use of our most precious resources - time, energy, attention and why not money, which ultimately translate into a better quality of life. Wikipedia[1] defines the PIM: "Personal information management (PIM) refers to both the practice and the study of the activities people perform in order to acquire, organize, maintain, retrieve and use information items such as documents (paper-based and digital), web pages and email messages for everyday use to complete tasks (work-related or not) and fulfill a persons various roles (as parent, employee, friend, member of community, etc.).".

In the light of this definition, we can easily see that a simple desktop search tool would offer the user only a partial solution. Whenever we work on our computers, we make use of various resources on the desktop, so just finding them on our PCs would not suffice, but a way to represent the context that we perform our taks into

[1] http://en.wikipedia.org/wiki/Personal_information_management

would be much more helpful. Imagine you want to find not only a document you saved from an email, but also the email that it was sent with. The idea is to link documents according to where they are used - to their working contexts. We are not only interested in documents per se, and to group them together, but we focus more on the very content of those documents and how some parts of them relate to each other, called semantic links, since they are related by meaning, not only by their physical appearance, as in the case of grouping files into folders. We propose various techniques in order to keep these semantic links between resources which are able to recreate working contexts on our desktops.

Most recent PIM projects focused on improving the management of desktop resources (e.g., files, emails), going beyond the functionality of commercial desktop search engines. Systems such as Haystack [QK04], Stuff I've Seen [DCC+03], and NEPOMUK [NEP] proved that automatically extracting and maintaining rich information describing desktop objects is feasible, but is not always used properly to easen the search task. The obvious shortcoming of generation of all this extra data is that they add up to the already huge size of the documents on our desktops, making it even more difficult to distinguish between the useful and needed resources. This is where Information Retrieval (IR) can play a role. In most existing cases, the way the user queries his information is by typing some keywords in a simple search box. The outcome is a long list of, hopefully, ranked results. Yet this also comes with a challenge: To fastly locate the needed information within those results. So a ranking is most needed here, but even so, just TFxIDF measures are not enough. We need to make good use of the generated connections between our desktop files. A semantic ranking, combining text properties with contextual properties seems more apropriate in this scenario, the solution we also propose in this thesis.

With the boom of the Social Web and online communities (*e.g. Del.icio.us*[2], *Facebook*[3], *YouTube*[4], *etc.*), we work more and more within communities. We communicate a lot daily, exchange ideas, pictures, documents, links, instant messages, and so on. So we can say we have reached a different level on our current desktops, the expanded desktop into a collaborative environment, represented by the desktops of our friends and family from where the exchanged resources come from and go to. Obviously, problems regarding privacy and trust arise immediately, since we have to be aware about the identity and intentions of people that we exchange resources with. We propose several solutions which try to cope with the trust and privacy problems - even if we trust some people enough to exchange resources with, we only want to share some of our resources and with certain restrictions. Also, when receiving resources, again a ranking mechanism needs to be in place, and also needs to take into account the trust that we have into the sender. Such ranking mechanisms for these collaborative environments are proposed in this thesis and supported with experiments in

[2]Delicious. http://delicious.com
[3]Facebook. http://www.facebook.com
[4]YouTube. http://www.youtube.com

order to show their performance and capabilities.

1.2 Open Challenges

We first list the main challenges addressed in this thesis and then explain each of them and show the proposed solutions within this thesis:

- **Challenge 1:** How to better exploit implicit linkage and semantic structures on the desktop?
- **Challenge 2:** How to help the user in overcoming the continoulsy growing amount of data available on the desktop?
- **Challenge 3:** How to support the user in sharing and protecting information?
- **Challenge 4:** How to interact with the Virtual Personal Information Space of the user?

People are daily faced with a lot of data coming from various sources which we need to manage. The solutions that we propose in this thesis recreates the links between the desktop resources in order to keep track of their semantic meanings, in order to make retrieval of these resources much easier. The solution of **Challenge 1** is a semantic desktop which keeps track of the semantic links between resources.

Since the number of resources that we have to deal with every day is becoming overwhelming, a sense of order needs to be in place, in order to be able to sort out the most important resources we have on our computers. A solution is proposed to **Challenge 2** in the shape of a semantic ranking mechanism, which exploits the created semantic links on the desktop.

The scenario of a single desktop proved to be insufficient into a world in which people interact all the time within communities. For the collaborative tasks which the user is faced with, we need some techniques in order to be able to filter out bad recommendations coming from people which we do not trust too much. **Challenge 3** is solved by proposing a collaborative ranking algorithm which is able to take into account the trust we have into certain persons and provide an appropriate ranking.

In a distributed environment, where the user's resources are not only located on his personal desktop, but also within the Web, we introduce the notion of the Virtual Personal Information Space of the user, which comprises all his resources, not necessarily on his desktop. In this environment, a continous exchange of resources takes place with other peers, where some of the resources are also shared in between several users. In this scenario, privacy and trust problems always arise, which we adress within **Challenge 4**. We propose a distributed ranking mechanism which is able to take into account the level of trust we have into a person and translate it into the rankings given to the resources coming from that person.

To sum up, in this thesis we envision and implement a desktop search tool capable of recreating "lost" connections on the desktop among its resources. These connections recreate at their turn working contexts to help the user, and also facilitate a ranking mechanism on top of the classic PageRank algorithm. Also, ranking solutions for the collaborative scenarios are proposed, also focusing on the reccomendations exchanged between users and the trust and privacy issues which need to be taken care of in such an collaborative environment.

1.3 Structure of the Thesis

The rest of this thesis is structured as follows:

In Chapter 2 (**Challenges 1 and 2**) we show our Beagle^{++} system, a fully deployed system, in which we created some modules capable of generating semantic information which is able to mantain the context connections for emails, web pages, files within directories and publications (Section 2.3). We use these links and suggest a ranking method for retrieved results and show our method performs better than the original system without our enhancements. Additionally we propose two methods for detecting contexts on the desktop. The first one (Section 2.4) exploits the temporal access relationships between the accessed files and builds on the main idea that if two files are accessed within a short time frame and rather often, then they are connected, since they might be used within the same task. The second one (Section 2.5) tries to directly detect contexts in a cluster fashion, by taking into account various evidences (text, folder hierarchies, access times, access in a sequence) translated into a Bayesian Network. The last proposed algorithm (Section 2.6) analyses the gain that an expanded personal desktop on the Web can give to the user. We generate from the desktop personalized annotations for visited Web pages, in this way transporting our working contexts from the desktop to the Web.

In Chapter 3 (**Challenges 3 and 4**) we focus more on the ranking part, where we first propose a ranking mechanism built on top of the semantic time links introduced in the previous chapter (Section 3.3). Next we focus on the scenario of collaborative desktops, where users exchange resources. We suggest a ranking algorithm for reccomendations between users and show how this can be easily adopted due to its good performance (Section 3.4). Also, when users exchange one or more resources, the trust into the other user influences a lot what we are going to handle and how we trust the resources coming from her. Therefore, we show how the level of trust into a person can be incorporated into the ranking mechanism, thus giving more importance to a resource coming from a more trusted person. In Section 3.5 we experiment with different levels of trust and also with different ways in which a user is willing to share his resources with others - totally or partially.

Chapter 4 summarizes the contributions of this thesis and discusses future ideas and problems for future work.

2 Generation of Desktop Context

2.1 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. With a few hundred of gigabytes at hand, it is quite common to have over 100,000 indexable items on the Desktop. It is no wonder that sometimes we cannot find a document anymore, even when we know we saved it somewhere. Ironically, in some of these cases nowadays, the document we are looking for can be found faster on the World Wide Web than on our personal computer. In view of these trends, resource organisation in personal repositories has received more and more attention during the past years. Thus, several research and development projects have started to explore PIM, including Stuff I've Seen [DCC+03], Haystack [QK04], or Gnowsis [SS04]. The PIM challenge is to make all resources on one's Desktop easily accessible and manageable. In this context, Desktop search is the obvious solution for finding such stored information.

In order to offer better results, current Desktop search engines have to improve the classic method of retrieval based on TFxIDF measures, and use additional information about the searchable resources. Currently, only few of the commercial Desktop search engines collect basic metadata, such as titles, authors, comments, *etc.*, usually already contained in the indexed files. However, since very few people spend time annotating their documents, this functionality provides only a limited improvement over regular text-based search. Studies have shown that people associate things with certain contexts [TAAK04], or to be more specific, everything happens within a context and a person will not think of a thing by its own, but within this very context. For example, a person will not only consider a document, but also the email that it was sent with and the person who sent it, *i.e.*, the context of the document. For this reason, this kind of information should be utilised during search. So far, however, neither has this information been collected, nor have there been attempts to use it.

In this chapter we propose to exploit the implicit semantic information residing

at the Desktop level in order to enhance Desktop Search. We therefore propose the automatic generation of metadata taking into account the context of Desktop resources:

- *Email context* clearly generates 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.
- *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 or comments. It would be helpful to find an attachment not only based on its content, but also based on its associated context[1] from within the email.
- *Folder hierarchies* may contain valuable context information, because we might have spent considerable time to build sophisticated structuring hierarchies for the documents we store.
- *Browser caches* include all information about the user's browsing behaviour. This is useful both for finding relevant results, and for providing additional context for them.
- *Downloaded publications* also miss all their "links", once stored on our machines. Yet it would be very useful if a search application not only returns one specific scientific paper, but all the referenced and referring papers which we downloaded on that occasion as well.

The additional metadata generated would be useless without a proper mechanism of querying and results ranking. Web search has become very efficient due to the powerful link-based ranking solutions such as PageRank [PBMW98]. The recent arrival of Desktop search applications, which index all data on the PC, promises to increase search efficiency on the Desktop. However, Desktop search engines are now comparable to first generation Web search engines, which provided full-text indexing, but only relied on textual IR algorithms for searching and ranking. We propose a centralised approach for querying, which combines the full-text and metadata search, and adds a modified ObjectRank [BHP04] mechanism for improved ranking of the retrieved results.

Through our extensions, we show that Beagle^{++} (the system we developed on top of the Beagle system) is not simply a *Semantic Desktop* (as Haystack [QK04], IRIS [CPG05], Gnowsis [SS04]), a PIM application (as SEMEX [DHN+04] or SIS [DCC+03]), or a new data storage paradigm (as Lifestream [FF95] or TagFS [BGSV06,

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

GSS06]). Instead, Beagle^{++} relies on a combination of all these aspects to provide a Desktop search engine that works on the "classic" Desktop metaphor and exploits the semantics contained in the Desktop data items. It is thus an example to illustrate the Semantic Desktop paradigm, demonstrating its benefits and potentials to ordinary users. Beagle^{++} is available for download[2] as sources, binaries and virtual machine.

The components making up Beagle^{++} contribute to the NEPOMUK project[3]. The goal of NEPOMUK is to create the Social Semantic Desktop which allows management of desktop resources as well as sharing and exchange of data between desktops [DF04, GHM^{+}07]. NEPOMUK provides an infrastructure for including various components in the Social Semantic Desktop application. All our components were also embedded in this framework, and thus also integrated with other components such as Gnowsis [SS04].

Even with all these tools at hand, the focus on the user's interests and working contexts is sometimes not very explicitly used. People tend to remember information in terms of associations and context [TAAK04]. Most research on context detection is incorporated in the broader field of desktop search systems, but not focused on context detection as a separate domain. In this work we argue that devising specialized algorithms for context detection would allow for a more efficient retrieval. Imagine that among the top-ranked documents of the results list of a search query, there is an email the user knows having sent in correlation with the information she is searching, but the email gives no clue of what or where the sought information is. Being able to retrieve the other documents belonging to the same context as the email would allow to find the searched information faster, thus saving time. This functionality allows us also to no more search for a precise document, but for one or more contexts, which reduces the search space, and allows us to save attention, since context is more natural than content.

So, the next obvious step in PIM research is to provide better task support on the desktop building upon the extracted information. Also, the borderline between desktop and Web deserves additional attention, since the personal information is no longer managed on the desktop only, but on the "Virtual Personal Desktop", an extension of the user's desktop on the Web. Our innovative approach for detecting (working) contexts serves this future PIM direction.

For the desktop, we define a context as the collection of resources that the user uses to solve one task. By processing these resources we collect similarity evidences, such as two files are considered to be related if they were accessed several times in a sequence. Also, identifying file similarities using text similarity techniques was found successful in many areas (e.g., [MdRA^{+}08] build on them for document classification). Our approach intelligently combines a variety of such evidences (textual and non-textual) to determine the working context. The resulting, improved knowledge about the user's context can, for example, be used for facilitating the user with the desktop

[2] http://beagle.l3s.de/
[3] http://nepomuk.semanticdesktop.org/

resources she needs for the current task as well as for targeted refinement of user profiles and, thus, for the personalization of Web search and recommendations.

More specifically, we focus on identifying evidences supporting possible file-to-context assignments, based on the content of the files on the desktop, as well as the time connections between them (e.g., files frequently accesed within a time frame, frequently accessed in a sequence). A Bayesian Network (BN) is created to model the evidences, the possible file-to-context assignments, and the interdependencies between them. We then use the BN to infer which files belong to which contexts.

As we already argued, the WWW has had a tremendous impact on society and business in recent years by making information instantly and ubiquitously available. The Desktop is now extended to the WWW, a vision of a future Web of machine-understandable documents and data. Annotation is seen as a means to enrich the Web with metadata. The "traditional" paradigm of Semantic Web (SW) annotation - annotating existing Web sites with the help of external tools - has been established for a number of years now, e.g., in the form of tools such as OntoMat [HS02] or tools based on Annotea [KKPS01], and the process continues to develop and improve.

However, this "traditional" paradigm is based on manual or semi-automatic annotation, which is a laborious, time consuming task requiring a lot of expert know-how, and thus only applicable to small-scale or Intranet collections. For the overall Web though, the growth of a Semantic Web overlay is restricted due to the lack of annotated Web pages. On the other hand, the tagging paradigm[4], which has its roots in social bookmarking and folksonomies, is becoming more and more popular. A tag is a relevant keyword associated with or assigned to a piece of information (e.g., a Web page), thus describing the item and enabling keyword-based classification of the information it is applied to. The successful application of the tagging paradigm can be seen as evidence that a lowercase semantic Web[5], comparable to the Web 2.0 ideas[6] – could be easier to grasp for the millions of Web users and hence easier to introduce, exploit and benefit from. One can then build upon this lowercase semantic web as a basis for the introduction of more semantics.

We believe that a successful and easy achievable approach is to automatically generate annotation tags for Web pages in a scalable fashion. With respect to *annotation*, we imply the use of a tag as a mechanism to indicate what a particular document is about (cf. [BM06]) as opposed to the use of a tag, for example, to organize the reading (e.g., "todo"). The possible drawback of automatically generated tags though, is that they present only one generic view, which does not necessary reflect any personal interests. For example, one user might categorize the home page of Anthony Jameson[7]

[4] http://en.wikipedia.org/wiki/Tag_(metadata)

[5] Lowercase semantic web is a term that seems to be coined by Tantek Çelik and Kevin Marks. It refers to an evolutionary approach for the Semantic Web by adding simple meaning gradually into the Web and thus lowering the barriers for re-using information.

[6] http://en.wikipedia.org/wiki/Web_2

[7] http://www.dfki.de/~jameson

with the tags "human computer interaction" and "mobile computing" because this reflects his research interests, while another user would annotate the homepage with the project names "Halo 2" and "MeMo" because he is more interested in research applications.

The crucial question is then how to automatically tag Web pages in a personalized way. In many environments, defining a user's viewpoint would rely on the definition of an interest profile. However, these profiles are laborious to create and need constant maintenance in order to reflect the changing interest of the user. Fortunately, we do have a rich source of information about the user available: everything stored on his computer. This *personal Desktop* usually contains a very rich document corpus of personal information which can and should be exploited for user personalization! There is no need to maintain a dedicated interest profile, since the Desktop as such reflects all the trends and new interests of a user, while it also tracks his/her history.

Based on this observation, we propose a novel approach for a scalable automatic generation of annotation tags for Web pages personalized on each user's Desktop. We achieve this by aligning keyword candidates for a given Web page with keywords representing the personal Desktop documents and thus the user's / author's personal interest, using appropriate algorithms. The resulting personalized annotations can be added on the fly to any Web page browsed by the user.

The contribution of this chapter is manyfold. We first show in Section 2.3a complete system - Beagle^{++}, a desktop search engine which we enhanced with metadata generators and a ranking scheme which exploits them. Then we show two methods for context detection on the desktop in Sections 2.4 and 2.5, and finally we argue that annotations are a good way to expand the desktop by annotating preferred web pages automatically with words picked from the personal desktop, is Section 2.6. With all these enhancements we try to model and expand the desktop in order to better reflect our interests and ultimately become a more helpful tool in our daily work.

2.2 Related Work

Desktop search applications are not new to the industry, only the high interest in this area is new: applications have been available since 1998 (*e.g.*, Enfish Personal[8]), usually under a commercial license. As the amount of searchable Desktop data has reached very high volumes and will most probably continue to grow in the future, the major search engines have recently given more focus to this area than the academia. Thus, several free Desktop search distributions have been released (*e.g.*, Google Desktop Search[9], MSN Desktop Search[10], *etc.*). Moreover, some providers have even in-

[8] http://www.enfish.com/
[9] http://desktop.google.com/
[10] http://toolbar.msn.com/

tegrated their Desktop search tool into the operating system, such as Apple[11]. The open source community has also manifested its interest in the area, the most prominent approaches being Gnome Beagle[12] (now also integrated into SuSE) and KDE KAT[13], developed within the Mandriva community. Other relevant commercial Desktop search applications exist, such as Copernic, Yahoo! Desktop Search, X1, Scopeware Vision, or PC Data Finder. Most of the above mentioned applications target a very exhaustive list of indexed file types, including any metadata associated with them. They also update their index on the fly, thus tracking changes on the Desktop. However, they either inherently miss the contextual information often resulting or inferable from explicit user actions or additional background knowledge, or they are limited to a small hard-coded set of metadata [DCC+03].

We will further present previous work developed in various research areas related to our tools and compare them to our present work.

2.2.1 Metadata Generation

Metadata Enrichment

Using Metadata to Enrich Search Results. One interesting semantic search tool that uses metadata to enrich search results is the TAP project [GMM03]. 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 web pages templates, being able to read and extract knowledge from several web sites. The key idea in TAP is that for specific searches, a lot of information is available in catalogues 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 IR technologies and aim to augment them, as opposed to our approach, where the semantic results are merged with the traditional ones.

In [NHCS07], the authors target to improve another domain, namely digital libraries. In this context, creating collections of metadata records from disparate and very diverse sources is a very tedious task, often leading to inaccurate, incomplete or even missing subject metadata. However, having proper subject metadata information is highly desirable, as this information enables users to more easily discover and browse documents by limiting the results based on their subjects matching the queries. The approach proposed in the paper, thus aims at improving the subject metadata quality by using statistical topic models, which can be also augmented with human review and intervention for filtering out the low quality topic labels, subject to be associated with the data's subject records.

[11] http://www.apple.com/macosx/features/spotlight/
[12] http://www.gnome.org/projects/beagle/
[13] http://kat.mandriva.com/

[LGZ08] adopts a totally different perspective regarding the use of metadata for enriching the search results. Here, the authors consider only metadata in the form of collaboratively created user tags and use this information for inferring users' social topic interests. The created user profiles can then be used to personalise search results, or connect like-minded users inside online social networks, such as Del.icio.us.

[CHSS08] is also making use of tags, though for a different setting, where tags are automatically attached to Web pages. Here the tags represent concepts extracted from a known set of concepts without any need of labeled documents and for achieving this, the authors propose a probabilistic modeling framework that combines both human-defined concepts and data-driven topics.

Using Metadata to Connect Information. In "The Social Semantic Desktop" [DF04], 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 with an email. Altogether, the entire information existing in a social network would be connected to each Desktop. Such a structure would then help people organise and find information, due to the enhancement brought by metadata into the system. We tend to follow this direction and enhance the resources on the Desktop with a lot of metadata, which finally translates into multiple connections between data items.

The Fenfire project [Fal04] 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 organise their data closer to the reality and to their needs, in which making comments and annotations would be possible for any file.

Haystack [QK04] pursues similar goals as Fenfire. One important focus is on working with the information itself, not with the program 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 [Sau03]. The main idea behind applications in this environment is the use of a central information server which allows users to manage and directly access all the information on their computer (for example the author of a file, her email address, *etc.*). Gnowsis envisions the possibility to link any two resources on the Desktop with a semantic connection.

In the context of another interesting prototype, the interface proposed by Yee *et al.* [YSLH03] 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.

As compared to all these previous approaches, we have the same aim of connecting resources located on the Desktop, but we further utilise these links in order to build a better search tool on the Desktop which combines the traditional IR methods with the semantic search, also allowing the user to better visualise this metadata network and browse it.

Metadata searching

Using Context Metadata to Find Information. Naaman *et al.* [NHW+04] describe an interesting approach for exploiting additional metadata for pictures retrieval. The idea is to rely on automatically generated metadata (location, time and other digital photo metadata) and manual annotations (events, *etc.*), automatically enhance these metadata by providing information about actual light status (night, day, dawn, dusk), weather conditions, temperature or additional aspects on the events, and then use these metadata to find stored images.

Another semantic search method using metadata is proposed by [RSdA04]. It first does 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 IR.

[WZ02] presents a new approach to content-based image retrieval. To improve the retrieval performance, the authors use a self-adjustable metadata store, which records the optimised 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, such as HSV colour-histograms or directional histograms.

Our approach focuses on a very wide range of metadata, not only low-level ones, which is generated fully automatically.

Querying. In the context of semantic querying, several languages which are used to interact with the repository have been proposed. The two most common languages to query RDF are SPARQL, a W3C recommendation, and SeRQL, which has been created for the Sesame repository. SeRQL is a language similar to, and in some means extending SPARQL. Their main characteristic is that it is possible to obtain an RDF graph as a query result. Both query languages also support named graph querying.

The disadvantage of these languages is that the user has to learn a complex query languages, therefore in Beagle^{++} we did not adopt any of these query languages for the user interface. In our case, the user will need either to type keywords in order to search for content, or to type queries in the format "property:value" (*e.g.*, "author:john"). The system then translates the user query into a suitable format for

the RDF repository.

Ranking. There are currently only limited (published) insights into the question of how to rank Desktop search results, mostly based on very simple techniques. Swoogle [DFJ+04] 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 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 users' preferences and practices. Our ranking algorithm resembles the method presented in [BHP04], where the authors apply authority-based ranking to keyword search in databases modeled as labeled graphs.

The importance of semantically capturing user interest is for example analysed in [AMHAS03]. 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 user interests and use this information to compute weights for the links among the semantic entities. In our system, the user interest is a consequence of her activities. This information is reflected in the properties of the entities defined. The weights for the links are defined manually.

2.2.2 Context Generation

Even if the implicit feedback that we receive from various tools was proved to be as accurate as other predicting methods about the importance that one resource has for its user, there is almost no work at all dedicated to using these importance measures to cluster the documents on one's desktop, and use them as working contexts (implicit feedback tools monitor user activities in an unobtrusive way, thus collecting information about the user's interests). We present previous work exploring text based clustering, but also try to give a broader overview of the measures that can be extracted about the activities of the user when benefiting of her personal resources.

Desktop Usage Analysis

Desktop usage behavior has been thoroughly analyzed in many studies. For example, Malone [Mal83] used interviews to analyze the way professional and clerical office workers organize information in their desks and offices. He identified two broad types of persons, *filers*, who organize their data into directories and categories, and *pilers*, who simply store all files in as few directories as possible. This work is orthogonal to ours, as we also analyze desktop user activity, but we focus on file access distribution, rather than storage behavior. Also, [BN95] suggested that the way information is used on the desktop should also be the primary determinant of the way it will be organized, stored and retrieved. We rely on the same idea: the user's way of interacting with

information on the desktop should deliver good evidences for identifying the user's contexts. Finally, [JDB02] investigates methods for organizing Web information for re-use, such as send email to self, print out the Web page.

Text Based Context Detection

[BHB01] uses text from productivity applications (like word processors, browsers, etc.) to extract keywords representative of the task the user is performing – i.e. the context. Those keywords are then used to pro-actively present the user with documents in relation with her current task, as opposed to our approach which is a static context detection, based on usage activity. Their type of clustering, based on similarity of results' titles and URLs, allows to group very similar documents, such that the user is not overwhelmed with numerous poor distinct results. We also perform text clustering but with standard algorithms, based on word vectors.

Scatter/Gather is a browsing method for results from a search, presented in [HP96]. It uses the Fractionation algorithm to perform text clustering on search results and automatically organizes them into a given number of topic-coherent groups. The user can then choose a cluster or a set of them to display only documents belonging to those clusters. Even though [CPKT92] presents an innovative browsing technique, it relies on the traditional word vectors which are used as a base for clustering. We could even think of using our new activity distance in a way for Scatter/Gather to take into account the time dimension.

[RMO+93] builds upon the idea that if a user organizes her desktop as a "filer", the type of organization can still be seen as a "piler", since the hierarchies built for storing the files on a computer are too complex, so the user would no longer be able to manually locate her data. They suggest using clustering based upon the term vectors of documents to help in reorganizing the resources already stored on the desktop, but also for giving hints where a newly created file can be classified. In our approach, we aim at clustering desktop resources for the purpose of detecting the contexts in which activities are performed on the computer, and use a different cluster identification technique, based upon the access behavior of the user.

An interesting idea on how text clustering can be used on the desktop comes from [Sta97]. They propose to exploit besides vector space models for lexical clustering, the idea of lexical chains, that is the possibility to identify the part where a context is active inside a text document. They also rely on the WordNet lexical reference vocabulary to disambiguate senses of words in different contexts. One of the described applications of their system, QUESCOT, tries to detect topic changes in documents but also, more interesting to us, how is a document relevant to a query from the topics perspective, by determining a context of occurrence for each query concept.

User Activity and Implicit Feedback

Although implicit feedback was proved to be as accurate as other methods for predicting the importance of a resource for its user, there is almost no work dedicated to using it to group the documents on one's desktop, and use them as working contexts as we do in our approach. In [OK01], a broad discussion has been made about the different types of behavioral patterns that can occur on the desktop. Their experiments relied on implicit feedback given by various tools, and mainly based their observations on analyzing the reading time durations, the time that the user spent on one resource. Of course that a more accurate measure would be to monitor attention focus only on specific units within documents, since for example, we would find the relevant background of a paper interesting and not the whole. Most of the examples given rely upon previous work performed in information filtering research, as they have proved that in general there is a strong correlation between the reading time and the importance of a document in the interests area of the user, since the recommendations based on this measure, were very accurate [MS94, KMM+97, CLWB01] (mostly done for browsing behavior on the web). This helps us in supporting our opinions, since the reading time is the difference between close and open time and we actually log and use these access times. Oard and Kim [OK01] conclude with the idea that behavior evidence is hardly taken into account in present research, and even less effort is put into combining this with content-based representation, the approach that we envision in this work.

All these observations were reinforced once again by almost the same type of studies performed in a non-laboratory environment and presented in [FKM+03]. They also demonstrated by their web search experiments that the time that a user spends on one document is the most relevant in order to predict a certain satisfaction, as in our case it will be able to predict a stronger appertaining to a context or another. As Fox et al. also mentioned, these observations can be further improved by logging the rare but very meaningful facts that occur on one's desktop, that the user also printed out a copy of the present document as it might mean it is more important to the user, as [KOR00] also suggested, and also that a certain link was added in the bookmarks. A good summarization on implicit feedback measures, categorization and usage can also be found in [KT03].

Implicit feedback is what [SG05] uses by analyzing file activities to deduce links between them based on temporal locality – i.e. when they were accessed. These links are then used to provide ranking using three different algorithms, namely Basic-BFS, HITS, and PageRank. We also use implicit feedback but to detect context instead of ranking.

2.2.3 Generation of Annotations

This work presents a novel approach, which makes use of document similarity and keyword extraction algorithms in order to generate personalized annotation tags for Web pages. Though blueprints for this approach exist, to our knowledge there has been no prior explicit formulation of this approach, nor a concrete application or empirical evaluation, as presented in this paper. Nevertheless, a substantial amount of related work already exists concerning the general goal of creating annotations for Web pages, as well as keyword extraction. The following sections will discuss some of the most important works in the research areas of annotation, text mining for keyword extraction, and keyword association.

Generating Annotations for the Web

Brooks and Montanez [BM06] analyze the effectiveness of tags for classifying blog entries. They find that manual tags are less effective for indicating the particular content of a document. We see this as a support for our work, because our evaluation (see Section 2.6.2) proves that the tags we create result in high precision for content description. They further show that cluster algorithms can be used to construct a topical hierarchy amongst tags. These findings could be a useful extension to our approach.

Cimiano et. al. [CHS04] propose PANKOW (Pattern-based Annotation through Knowledge on the Web), a method which employs an unsupervised, pattern-based approach to categorize an instance with respect to a given ontology. Like our approach (denoted as P-TAG hereafter), it is rather simple, effortless and intuitive to use for annotating Web pages. However, for this kind of annotation, PANKOW requires an ontology and annotates a Web page with instances of the ontological concepts, whereas we annotate Web pages with user specific tags. Also, PANKOW exploits the Web by means of a statistical analysis, hence the annotation reflects more common knowledge without considering context or personal preferences, while in our approach, we create personalized tags. The main drawback of PANKOW is that the approach does not scale, since it produces an extremely large number of queries against the Google API.

The work in [CLS05] presents an enhanced version of PANKOW, namely C-PANKOW. C-PANKOW downloads abstracts and processes them off-line and thus overcomes several shortcomings of PANKOW. Furthermore, it introduces the notation of context, based on the similarity between the document to be annotated and each of the downloaded abstracts. However, it is reported that annotating one page can take up to 20 minutes with C-PANKOW. Our system annotates Web pages on the fly in seconds. But the tasks are not entirely comparable, since our system doesn't produce ontology-based annotations, but personalized annotation tags. Further, our notion of context is much stronger, since we consider documents from the personal

Desktop, which leads to highly personalized annotations. Finally, C-PANKOW uses the proper nouns of the Web page for annotation candidates, and thus annotation is always directly rooted on the text of the Web page. On the other hand, the algorithms we propose in this work generate keywords that not necessarily appear literally on the Web page, but are in its context, as well as in the personal interest of the user.

Dill et. al. [DEG+03] present a platform for large-scale text analytics and automatic semantic tagging. The system spots known terms in a Web page and relates it to existing instances of a given ontology. The strength of the system is in the taxonomy based disambiguation algorithm. Our system does not rely on such a handcrafted lexicon and extracts new keywords in a fully automatic fashion, while also supporting personalized annotations.

Text Mining for Keywords Extraction

Text data mining is one of the main technologies for discovering new facts and trends about the currently existing large text collections [Hea99]. There exist quite a diverse number of approaches for extracting keywords from textual documents. In this section we review some of those techniques originating from the SW, IR and Natural Language Processing (NLP) environments, as they are closest to the algorithms described in this work. In IR, most of these techniques were used for Relevance Feedback [Roc71], a process in which the user query submitted to a search engine is expanded with additional keywords extracted from a set of relevant documents [XC96]. Some comprehensive comparisons and literature reviews of this area can be found in [Eft95, VWMFC05]. Efthimiadis [Eft95] for example proposed several simple methods to extract keywords based on term frequency, document frequency, etc. We used some of these as inspiration for our Desktop specific annotation process. Chang and Hsu [CH98] first applied a clustering algorithm over the input collection of documents, and then attempted to extract keywords as cluster digests. We moved this one step further, by investigating the possibilities to acquire such keywords using Latent Semantic Analysis [DDL+90], which results in more qualitative clusterings over textual collections. However, this turned out to require too many computational resources for the already large Desktop data sets.

The more precise the extracted information is, the closer we move to applying NLP algorithms. Lam and Jones [LAJ01] for example use summarization techniques to extract informative sentences from documents. Within the Semantic Web / Information Extraction area, we distinguish the advances achieved within the GATE system [CMBT02, MBC03], which allows not only for NLP based entity recognition, but also for identifying relations between such entities. Its functionalities are exploited by quite several semantic annotation systems, either generally focused on extracting semantics from Web pages (as for example in KIM [KPO+03]), or more guided by a specific purpose underlying ontology (as in Artequakt [AKM+03]).

Text Mining for Keywords Association

While not directly related to the actual generation of semantic entities, keyword association is useful for enriching already discovered annotations, for example with additional terms that describe them in more detail. Two generic techniques have been found useful for this purpose. First, such terms could be identified utilizing co-occurrence statistics over the entire document collection to annotate [KC99]. In fact, as this approach has been shown to yield good results, many subsequent metrics have been developed to best assess "term relationship" levels, either by narrowing the analysis for only short windows of text [GWR99], or broadening it towards topical clusters [WT06], etc. We have also investigated three of these techniques in order to identify new keywords related to a set of terms that have been already extracted from the Web page which requires annotation. Second, more limited, yet also much more precise term relationships can be obtained from manually created ontologies [CHBG01], or thesauri, such as WordNet [Mil95].

2.3 The Desktop Search Beagle^{++} System

2.3.1 Enhancing the Beagle Desktop Search Architecture to Support Metadata — An Overview

As basis for our Beagle^{++} environment we use the open source Gnome Desktop search engine Beagle[14] for Linux, which we extend with semantic indexing, searching and ranking capabilities. The reason for choosing Gnome Beagle to build upon was to reuse existing work on developing and establishing a Desktop indexing and searching platform, such that we could primarily focus on developing the semantic part of our Semantic Desktop Search engine.

Figure 2.1 illustrates the overall Beagle^{++} architecture. For maintaining the generated information up-to-date even if the physical files on the Desktop change, we rely on the *inotify*-enabled Linux Kernel, which catches all system events, *i.e.*, files being created, modified or deleted. All these kinds of Desktop events are sent to the *Beagle Server*, which is in charge of dispatching, indexing and sending requests to the appropriate module. Several modules ensure the execution of the following processes supported by our architecture:

- The extraction of metadata is issued by the *Metadata Extractor Backends*, which dispatch extraction tasks to the appropriate filter components, according to the type of resource and content to be extracted. Information about the content of resources is extracted using the *Content Filters* which act on specific file types (*e.g.*, PDF, XLS, DOC, EML, *etc.*[15]). The *Metadata Filters* extract specific

[14] http://beagle-project.org/
[15] http://beagle-project.org/Supported_Filetypes

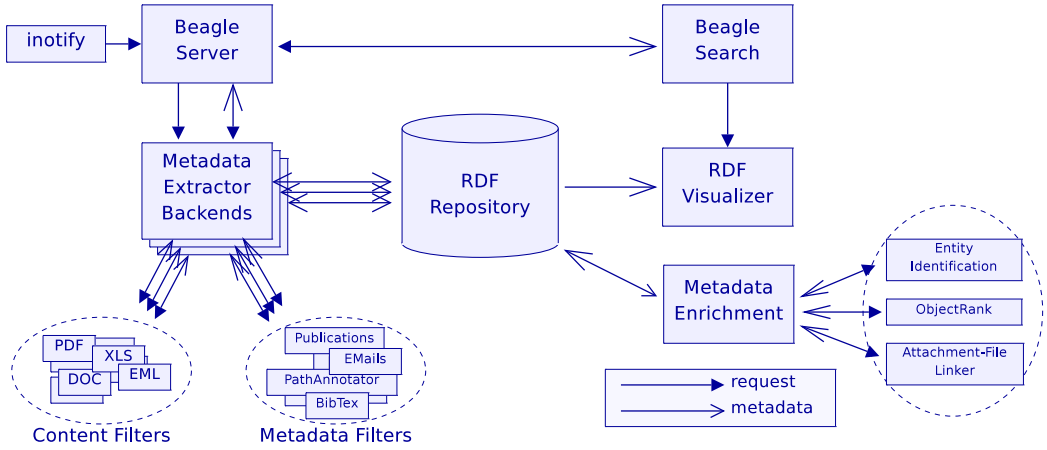

Figure 2.1 Beagle^{++} Architecture Overview

information (Publications, Emails, BibTeX, Web pages), according to the file's role in the user activities, for example, the authors, title and citations will be extracted from a publication.

- The enrichment of these metadata is performed by three *Metadata Enrichment* modules we developed: *Entity Identification*, *ObjectRank* and *Attachment-File Linker*.

- The extracted content and metadata information are stored and indexed in the central *RDF repository* .

- The search part of our Desktop search engine is provided by the *Beagle Search* module. It is also responsible for routing the users' search requests to the *Beagle Server*, which further hands them over to the *Metadata Extractor Backends*. These try to find relevant results matching the queries in the RDF repository. The search results coming from the *Metadata Extractor Backends* are merged by the *Beagle Server* into one list of result documents, which is presented to the user by *Beagle Search*.

- For being able to exploit the added value of the generated metadata, the results should be visualised in an adequate way to the user. We therefore created the *RDF Visualiser*, which supports *Beagle Search* in presenting the search results by visualising the RDF graph around matching documents.

Related projects like Haystack or Gnowsis similarly employ extractors, each one specialised for one type of information source. The extracted semantic information are represented in RDF and stored in a centralised and uniformly accessible point. Therefore, such an architecture has proven to serve our needs to develop a high quality semantic retrieval system.

In the following sections we detail some of the components present in our Beagle^{++} architecture, by showing the provided functionalities and techniques behind them. For a complete description of all the modules, please refer to [MPC+10].

2.3.2 Metadata Generation and Storage

As already presented in Section 2.3.1, an important functionality of our Beagle^{++} Desktop tool is the creation and storage of metadata. Since these metadata are used and processed by an extensible set of components, compliant with a common well-defined ontology, such that every component which generates and consumes metadata can rely on their format and semantics. In the following, we will describe our metadata generation modules.

Metadata Extraction

With the help of the ontologies described in [MPC+10], we can represent metadata extracted from Desktop data sources such as files, emails, contacts and calendar items, instant messaging logs, notes, Web history, to name only a few. For each data source, one of the Metadata Extractor Backends introduced in Section 2.3.1 is in charge of indexing its data items. These backends are processing, for example, the file system, an email client's inbox or an instant messaging program's log files, the corresponding data items being files, emails and instant messages, respectively.

For each data item to be indexed, an appropriate *Filter* for processing the content and the metadata is selected. The extracted information is stored in the *RDF Repository*. Each *Filter* processes one specific type of file, identified by filename extension or MIME type (*e.g.*, *.pdf* or *application/pdf*, respectively). Based on the type of extracted information, the Beagle^{++} Filters are classified into two categories: *Content Filters* and *Metadata Filters*. The Metadata Filters are specific to Beagle^{++} and are not present in the original Beagle architecture.

Our Metadata Filters improve existing Desktop search systems by extracting new and enhanced metadata information from four specific sources. In the following, we describe these sources in detail:

File Paths. Folder hierarchies are barely utilised by existing search algorithms, in spite of the often sophisticated classification hierarchies users construct. For example, pictures taken in Hannover could be stored in a directory entitled "Germany", so it would be useful to use this information for search. Normally, a simple search for "Hannover" would be unable to retrieve the pictures in that folder. The PathAnnotator component annotates files with every token appearing in their file path, as well as additional semantic information provided by the WordNet system[16], such as direct synonyms, hyponyms, hypernyms, meronyms

[16] http://wordnet.princeton.edu/

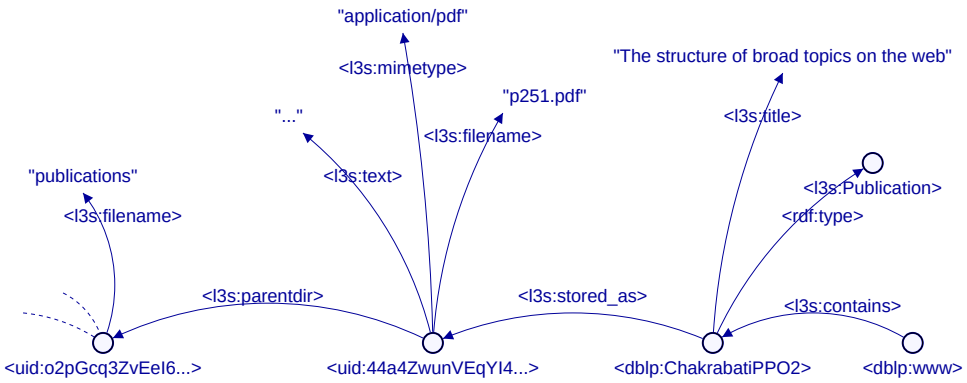

Figure 2.2 An example metadata graph extracted from one data item (here a PDF file representing a publication).

and holonyms [CGG+05], for each of these tokens, but naturally excluding the trivial tokens as "/home/". This kind of additional information is particularly important since for specific types of files, which do not provide abundant textual information, the search process can rely upon the *text* generated for this resource, based on the effort that the user invested to organise her files in a hierarchy. For example, a picture stored in the folder "Germany" is annotated with the term "Hannover", because Hannover is a part of Germany. Then, this pictures can easily be retrieved by searching for the term "Hannover".

Email Metadata. Emails provide much structured information about people and how people and emails relate to each other. For instance, an email can be sent as a reply to another email, where each email is globally uniquely identified. People that send or receive an email are also uniquely identified by their email addresses. The inbox and the "sent" folder of an email client is a rich source for such structured metadata. With our Metadata Filter for emails, all this information becomes explicit in the RDF repository. Due to the unique identifiers for emails and people, small graphs extracted from each email can be integrated into a large network of metadata.

Web Cache. The linkage information between web pages is of high value on the Web for various techniques such as ranking. To have the same degree of connection on the Desktop, we broadened the notion of a "visited link" by defining it as a web page that was previously visited by the user (*i.e.*, the link's target page is present in the browser cache). The WebCache component transforms the visited links into metadata which are created for every web page in the cache, representing the links that have been visited from that page, as well as the in-going links from which the user could have arrived to it (inverse of a visited link). This information is also of high value in order to connect the information on the desktop in a similar manner as on the Web, allowing for additional techniques, *e.g.*, ranking, but also the visited web pages become of a

high importance if we think of them as very useful in a given working context, where the user of the desktop required browsing for solving a certain task. In addition, these visited links are also highlighted in the Web browser, in order to facilitate navigation for the user. When searching for a web page, the user can reconstruct his previous navigational steps from another familiar web page.

Scientific Publications. In the research community, many papers are available in PDF format, but although PDF allows basic metadata annotations like title and authors, these are rarely used. We therefore developed a Metadata Filter which extracts metadata from PDF scientific publications and provides it to improve Desktop search. Since information extraction is a difficult and error prone task, we decided to leverage the publicly available DBLP[17] database. The Metadata Filter first extracts the title (as the most selective element for a publication) from the text of the PDF using different heuristics refined over a set of experiments[18]. The obtained title is then used to search the DBLP databases whose publication titles have been previously indexed using standard IR techniques. If the first ranked publication has a matching score above a predetermined threshold, it is assumed to be the publication contained in the PDF file, and publication metadata like authors, conference, year of publication, *etc.*, are retrieved from the DBLP database.

Storing and Indexing Metadata

For each data item a metadata fragment is created, which itself corresponds to a new data object, which needs to be stored and indexed for later search. Figure 2.2 presents an example of metadata extracted from a PDF file representing a publication. Along with the explicit metadata associated with the file (*e.g.*, file name, file size, creation and modification time, MIME type), we also extract more complex metadata, such as the folder where the file resides, title of the publication it contains, further referring to the conference it was published at, *etc.* The more of these metadata fragments become available in the RDF repository, the more Desktop resources get interconnected. For instance, all PDF files that contain publications being published at the same conference form a connected network around the respective conference. Eventually, all these metadata fragments get integrated into a large network inside the RDF repository.

For storing the produced metadata we employ the NEPOMUK RDF Repository developed in the NEPOMUK project [GHM+07]. Being based on the open source Java RDF storage, querying and reasoning framework Sesame [BKvH02], as well as on Lucene, which is incorporated into the Sesame framework via the LuceneSail [MSG+08], it benefits from the advantages of both: Sesame allows fast structural

[17] http://dblp.uni-trier.de/

[18] The heuristics include information like "the title is at the top of the first page, in bigger charachters".

(RDF) queries, and LuceneSail facilitates fast full-text queries over all literals, including the actual content of the data item (see Section 2.3.4). Besides, by using the capabilities of the Lucene inverted index, the RDF repository allows performing full text search in the literals of the generated metadata fragments, whereas using the Sesame store allows the system to perform structured search via the SPARQL[19] and SeRQL language[20]. Recent performance studies showed that the Sesame and LuceneSail infrastructure is very competitive compared to other available open source solutions [MSN09].

2.3.3 Metadata Enrichment

As already mentioned in Section 2.3.1, once the metadata are stored in the RDF repository, we propose to further apply several methods for enriching them. In this section we describe in detail these methods and more precisely two different modules encapsulating them: the *Attachment-File Linker* which preserves the links between emails and their attachments stored on the Desktop, thereby improving the retrieval effectiveness by creating more relations in the RDF repository, and, the *ObjectRank* module which adds metadata for supporting ranking the Desktop entities based on their link structure. In the following we will discuss in detail each of these modules.

Resource Linkage using Attachment-File Linker

To create new relations between Desktop resources we exploit users' actions on the Desktop objects. When performing a task, the user accesses various resources, which can be considered related. Therefore, we semantically translate the user activities on the Desktop into metadata. In this work, we translate such a user action: saving the attached files from an email to a folder on the computer. We thus aim at preserving the semantic connection between the attachment and the email it was sent with.

As soon as we save an email attachment on our disk, it becomes a simple file and the original connection to the email is lost. Imagine searching for an institution, L3S for example, and receiving as a result a document which apparently has no connection to the research lab. But, while browsing its RDF metadata, the user could see that it was an attachment to a file sent by a researcher in L3S, then, she could instantly remember the whole context of the discussion in the email and why that document was sent to her. An example of the generated metadata for such an email can be seen in Table 2.1.

In order to serve this kind of situation, we created a new metadata enrichment tool, the *Attachment-File Linker*, representing a linker between an email and its attachment saved on the disk. This basically adds an additional link in the RDF graph between

[19] http://www.w3.org/TR/rdf-sparql-query/
[20] http://www.openrdf.org/doc/sesame/users/ch06.html

Subject	Predicate	Object
file://mail1457.eml	rdf:type	Email
file://mail1457.eml	...from	mailto:firan@l3s.de
file://mail1457.eml	**...email_attach**	file://mail1457.eml/;SEC=0
file://mail1457.eml/;SEC=0	...file_name	WebLinks.pdf
file://mail1457.eml	...email_text	This is a good publication...

Table 2.1 Metadata generated for an email and its attachment.

a file and an email attachment, if the file was saved from the attachment of that particular email.

The application searches over the whole RDF graph (reading from the RDF repository) for two resources, one of type File and one of type Attachment, with the same size and extension, the date of creation of the file after the date of arrival of the email (that the attachment belongs to), and with similar names (we used the SecondString library[21] and a threshold of at least 0.5). The logic behind this type of search is strictly connected to the normal behaviour of a user who would save a file from an email, and would not change the extension of the file, or modify the file by itself or even drastically modify the name of the attachment.

In this way we can connect emails with the related attachments saved on the disk creating more semantic relations in the RDF repository, which helps us to improve the search effectiveness, as it better simulates the structure implicitly present on the Desktop. An additional component which exploits the semantic relations created by the previous resource linkage modules, will be presented in the next section.

Metadata Enrichment using ObjectRank

The main problem on the Desktop is that PageRank-like algorithms cannot be deployed successfully as long as the Desktop resources are not linked to each other. Therefore, what we need is a way to explicate the links among them and a way to use this link information in ranking. In addition to the email attachment links, [BCC+06] describes further services for creating explicit links between resources. A number of relationship types or property types are used to describe the relationships among the resources and thus influence the rankings.

ObjectRank [BHP04] has introduced the notion of *authority transfer schema graphs*, which extends ontologies, by adding weights and edges in order to express how importance propagates along the relationships inside an ontology. In our desktop search framework ObjectRank relies on ontologies (as the one presented in Figure 2.3) for modeling the importance flow among desktop entities. For example, the authority of an email is split among the sender of the email, its attachment, the date when it

[21] http://secondstring.sourceforge.net/

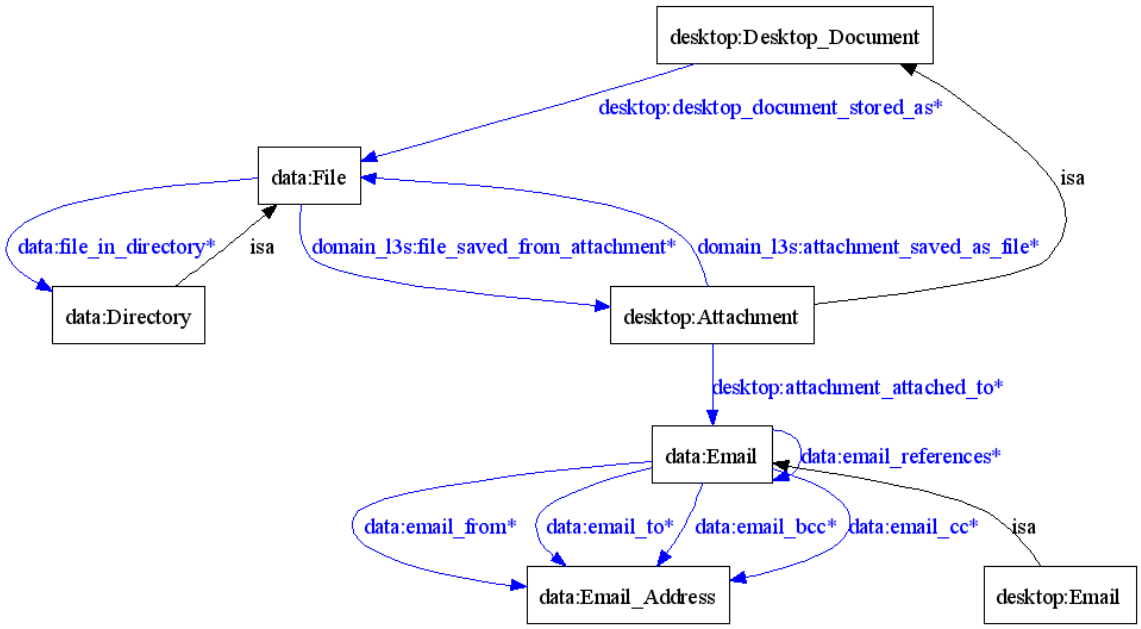

Figure 2.3 Desktop Ontology

was sent and the email to which it replied to. So, 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. A part of the Desktop ontology used for Beagle^{++} is depicted in Figure 2.3.

As suggested in [BHP04], in preparation of the computation, every edge from the schema graph is split into two edges, one for each direction. This is motivated by the observation that authority can potentially flow in both directions and not only in the direction that appears in the schema, *e.g.*, if we know that a particular person is important, we also want to have all emails we receive from this person ranked higher. With Desktop resources linked to each other, the final ObjectRank value for each resource is calculated based on the PageRank formula:

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

applying the random surfer model [PBMW98] 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. A is the adjacency matrix which connects all available instances of the existing context ontology on one's Desktop. Parameter d represents the dampening factor and is usually considered 0.85.

The ranking approach in Beagle^{++} relies on the metadata generators which store metadata information in the RDF repository. In order for ObjectRank to perform the

computation of the scores, all metadata that are stored in the repository have to be represented as triples, in the form Subject-Predicate-Object, and be compliant with the Desktop ontology. When instantiating the Beagle^{++} ontology for the resources existing on the users' Desktop, the corresponding matrix A will have elements which can either be 0, if there is no edge between the corresponding entities in the data graph, or have the value of the weight assigned to the edge in the authority transfer schema graph divided by the number of outgoing links of the same type. The ranking computation is performed offline, and the results are stored back into the RDF repository. Afterwards, the ObjectRank recomputation is performed every N minutes (configurable), and is triggered by specific scripts. All entries, which are present in the RDF repository are used for building up the data graph on which the ranking computation is performed.

2.3.4 Metadata Search

After populating and indexing the metadata store we provide the user with the search functionality as well as with the possibility to visualise retrieved Desktop items together with their metadata. In this section we will describe how Beagle^{++} performs the search, how the final ranking of the results is computed, and how it displays the retrieved resources.

Querying

When the indexing process is finished, the information about processed Desktop items is stored in the RDF repository (see Section 2.3.2) and it is possible for the user to query it. Beagle^{++} allows two types of queries: 1) usual keyword queries that only refer to the content of the indexed resources, and 2) combined text and metadata queries where specific keywords match the content and others match specific metadata values (*e.g.*, "text:technical metadata:lucene").

In contrast to other semantic search approaches like [TCRS07, WZL+08, ZWX+07], where pure keyword queries are given by the user and thus semantic information are implicit, in Beagle^{++} the user expresses semantic relations explicitly. This generally leads to the limitation that such semantics cannot be used for enhancing the search process unless the user knows how the structure of the metadata graph (the underlying ontology) looks like, that is, how the RDF properties are named. To cope with this problem, we manually established mappings between RDF property names appearing in the ontology, and values which the user may employ for those properties. For example, if the user would use the query "author:john" the system will map the property name "author" with http://beagle2.kbs.uni-hannover.de/ontology/publications#author, which appears in the underlying ontology.

We further describe in more details the querying process which takes as input the user query and provides back a ranked list of results (see Figure 2.4).

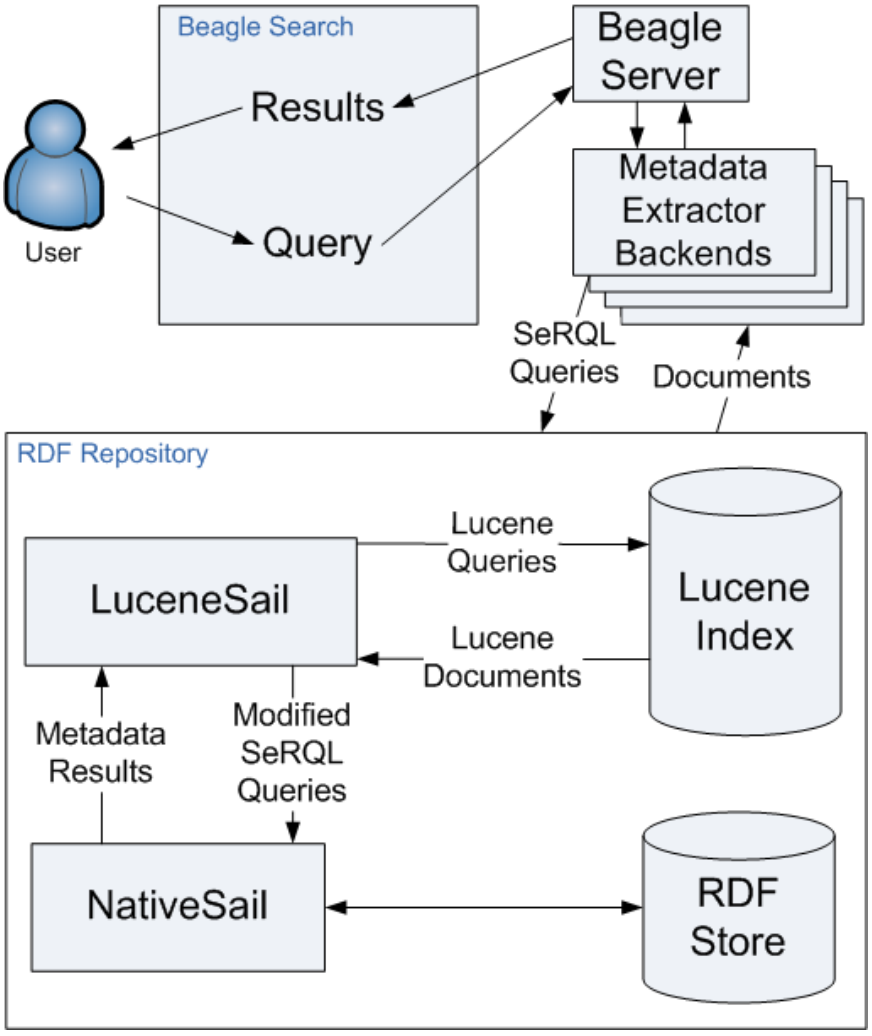

Figure 2.4 Detailed query processing in Beagle^{++}.

The user issues a query (for example "Spain") by typing it into the Beagle^{++} GUI, `Beagle Search`, which then sends it to the *Beagle Server*, and this one calls the backends (see Section 2.3.1) for obtaining different types of results. The user query is translated into a structured query and sent to the LuceneSail in order to be answered using both the Lucene Index and the RDF repository (see Section 2.3.2). LuceneSail uses the incoming queries to retrieve results (*i.e.*, Lucene Documents) from the Lucene Index and metadata results from the RDF repository via the NativeSail, which is in charge of retrieving RDF triples. Finally, the retrieved triples are converted to Documents in the standard Lucene format. In our example all information (both full-text and metadata) about "Spain" will be retrieved, including even the ones that are not explicitly about Spain, but describing for example Andalusia as a region of Spain. This is possible due to metadata extraction (see Section 2.3.2), which enriches resources with additional, related information.

LuceneSail merges the full text search in the RDF graph literals stored in the Lucene index (these RDF graphs are indexed as full-text and stored together with the content of resources) with the search for concepts and relations (for example, Andalusia is a region of Spain) in the native RDF store, enabling a structured and semantic search like we illustrated in our example.

At this point the search for items similar to the relevant ones is also performed. The list of relevant resources is expanded by searching in the RDF graph for resources linked via a property which indicates the similarity (see [MPC+10] for more details). For example, a misspelled name is similar to the correct one which will be also added to the results. The final list of documents is returned to be visualised by `Beagle Search` as described further in this Section.

Ranking in Beagle^{++}

The ranking schema we designed for Beagle^{++} uses a combination of the TF×IDF score returned by the LuceneSail and an extension of PageRank: *ObjectRank* (see Section 2.3.3). The motivation for such an approach is that users might want to find a specific document not only based on its content, but also based on the contextual information around it. Studies have shown that users tend to associate things to different contexts [TAAK04], which means that all this additional information should be utilised during search.

Let us now consider a scenario for validating our assumptions about the benefits of a search engine enhanced with ranking. We assume that Alice is a team member of a computer science research institute, and one of the topics she is interested in is recommender systems. Alice is currently writing a report about new techniques for recommending multimedia content and therefore needs to write a section summarizing the state-of-the-art of recommender systems. She remembers that she has stored on her Desktop a few good papers on collaborative filtering techniques, partially papers found on the web, partially received by email from different colleagues, and partially papers for which she attended the presentations at several conferences. For finding those papers, Alice uses a Desktop search engine, where she issues the query "collaborative filtering" and in response to her query she receives a long list of results, which unfortunately does not contain in the top-5 results the papers she was looking for. She would like to have instead a Desktop search engine which takes into account her personal preferences and presents the search results ranked based on this information.

The new ranking schema we developed benefits both from the advantages of Lucene's TF×IDF score and those of ObjectRank. The new scores are computed as a combination of them using the following formula:

$$\mathrm{R}'(a) = \mathrm{R}(a) \cdot \mathrm{TFxIDF}(a), \tag{2.1}$$

where a represents the resource, $\mathrm{R}(a)$ is the computed ObjectRank, $\mathrm{TFxIDF}(a)$ is the

TF×IDF score for resource a and R′(a) is the resulting score. The formula guarantees that the highest ranked resources have both a high TF×IDF and a high ObjectRank score. The re-ranking is performed at query time.

Coming back to the presented search scenario presented, we can see in Figure 2.5 and 2.6 how the order of the search results differs when using Beagle and Beagle^{++}, respectively:

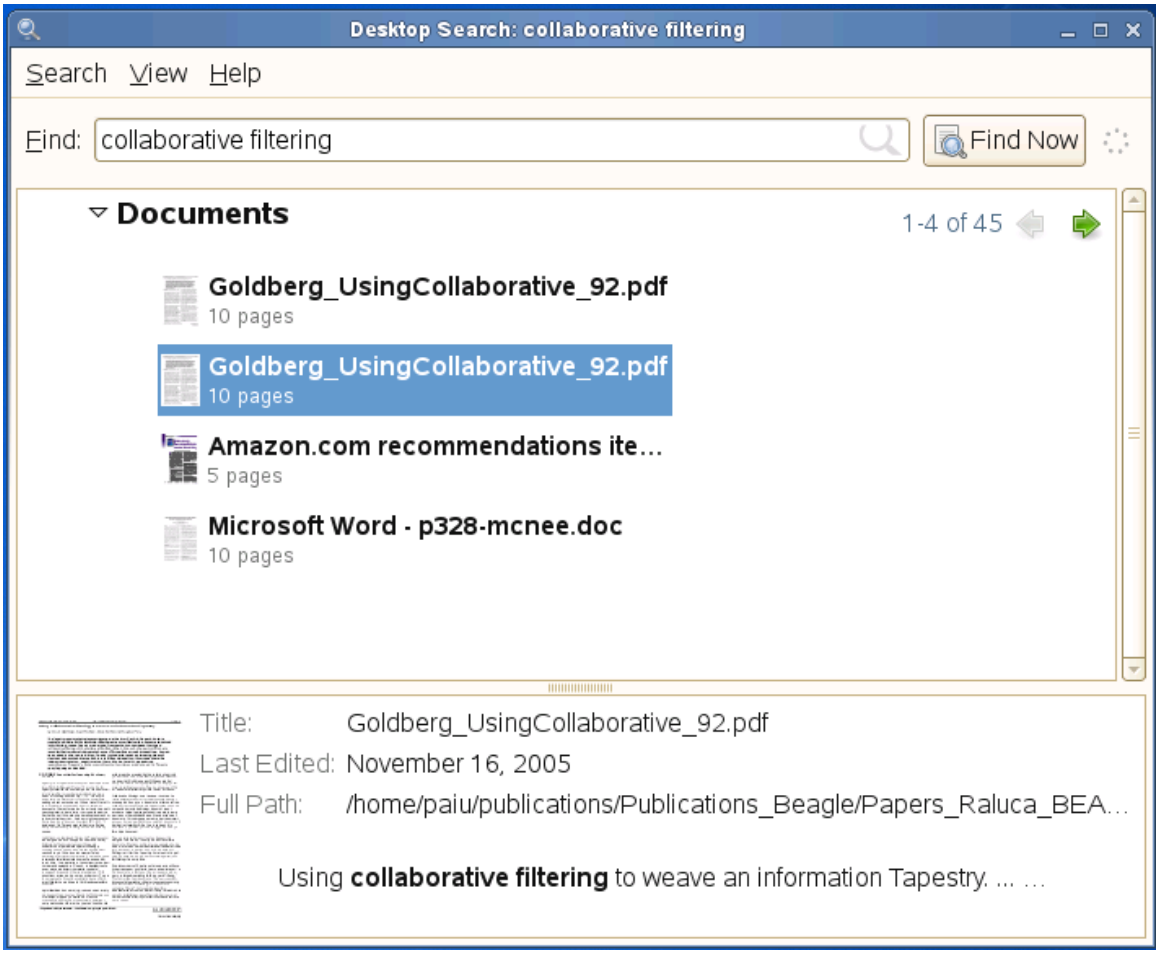

Figure 2.5 Search results retrieved by Beagle

When using the Beagle system (see Figure 2.5), Alice receives a list containing 45 results (also including duplicates) and the first ranked result represents a paper by Goldberg *et al.* — "Using collaborative filtering to weave an information Tapestry" — so, obviously a relevant paper. The second and third ranked results are papers on recommendations, however not very relevant ones.

The list of results Alice receives when using the Beagle^{++} system (see Figure 2.6) is shorter (duplicates are removed) and better ranked: as the first ranked result, this list contains also Goldberg's paper, but the next results are much more relevant for Alice's interests. The second best result is now a paper, which was accepted

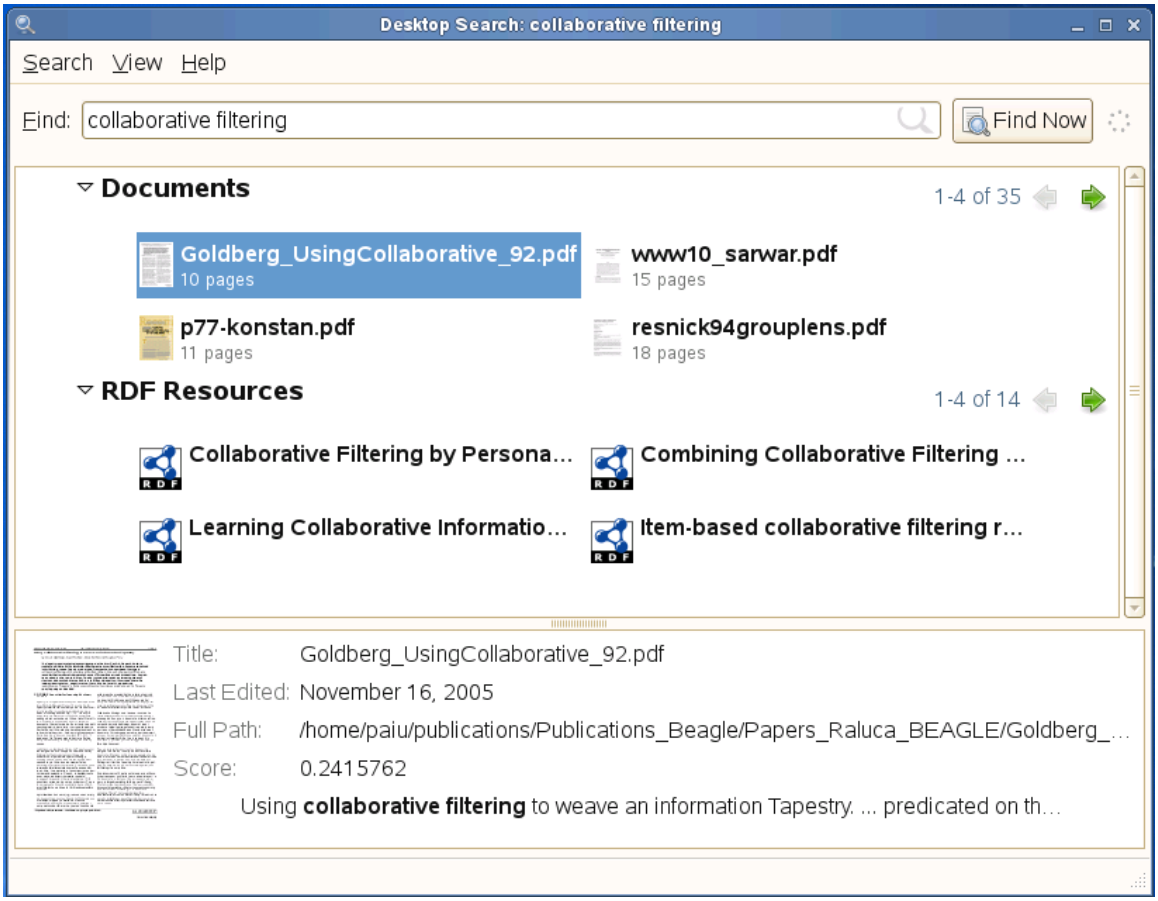

Figure 2.6 Search results retrieved by Beagle^{++}

at a WWW conference: "Item-based Collaborative Filtering Recommendation Algorithms" by Badrul Sarwar, George Karypis, Joseph Kostan and John Riedl. This conference is very important for Alice because she has stored on her Desktop several papers which were accepted at different editions of the conference. The metadata extraction phase could therefore find many links between all the papers published at that conference, their authors, and other papers and their authors that are referenced or also stored in Alice's Desktop. Prominent conferences and people of a certain domain are therefore more likely to have many links to conferences, papers and authors of their domain. This information is taken into account by the Beagle^{++} ranking so that highly connected authors, conferences or papers are higher ranked. The next hits in the list represent papers written by prominent people of the domain and at the same time very often cited in the literature.

Displaying Desktop Resources

After the data are indexed, queried and ranked, the results have to be presented to the user in a suggestive way within a visualisation interface. Since our main goal is improving Desktop search by exploiting networks of metadata, we did not focus on coming up with a completely new user interface, but rather took the *Beagle Search* interface and showed how to incorporate the browsing of a network of metadata into its task of displaying the found Desktop resources. Since this is not the focus of this thesis, we refer the reader to [MPC+10] for more details.

We have so far explained the modules composing the Beagle^{++} architecture and their functionalities: metadata generation, metadata enhancement, indexing and storage of data and metadata, search and visualisation of results. We further show how our Semantic Desktop search solution improves the quality of Beagle within a set of experiments.

2.3.5 Experiments

In order to evaluate the performance of our Beagle^{++} system, the natural baseline we considered was Beagle, since Beagle^{++} is an extension of Beagle. The first category of experiments aimed to prove the quality of the results provided by Beagle^{++}. It was done involving human judges who rated the results that our system provided to personalised queries. The second type of experiments considered the performance in terms of time to index collections of data, the amount of extra data (metadata) generated and the response time for queries. Both sets of experiments were conducted on 12 data sets, which consisted of the data provided by our researcher colleagues on a voluntary basis: emails, documents, publications, address books, calendar appointments and other resources found on the users' Desktops (.txt, .doc, .ppt, .html, *etc.*).

Data Set Description

For proving the quality and the performance of our Beagle^{++} system, we need a data collection which accurately represents Desktop data characteristics for testing our algorithms on. However, given the privacy concerns the users usually have when giving away their Desktop data — most of the times highly personal —currently there are no Desktop data collections publicly available. Moreover, testing algorithms on artificial datasets can be misleading and hard to evaluate as well. To overcome these problems and also to make our experiments repeatable, we compiled for experimental purposes a new Desktop data collection. More explicitly, we gathered real Desktop items from a number of 12 colleagues corresponding to emails (sent and received), publications (saved from email attachments, saved from the Web, authored / co-authored), address books and calendar appointments. The distribution of the Desktop items collected

from each user can be seen in Table 2.2.

User ID	Emails	Publications	Address books	Calendars
1	30,627	0	0	0
2	4,423	869	1	1
3	833	236	0	0
4	3,820	266	1	0
5	2,012	112	0	0
6	217	28	0	0
7	218	95	1	0
8	0	236	1	1
9	1,034	31	1	0
10	1,068	157	1	0
11	1,167	426	0	0
12	48	452	0	0
Total	45,467	2,908	7	2

Table 2.2 Desktop test data — Resource distribution over the users.

A total number of 56,484 Desktop items has been collected, representing 8.1GB of data, on average each user providing 4,707 items. The users provided a dump of their Desktop data, including all kinds of documents, not just emails, publications, address books or calendars. We have only included these types of specific resources in the above table, since they are the most important for our modules.

User Studies and System Quality

We first did an evaluation of our Desktop search engine by conducting a small scale user study[22]. Our colleagues, who provided us a subset of their Desktop data, had to define their own queries, related to their activities, and then performed searches over the above mentioned reduced dumps of their Desktops. Each user had to specify 8 queries in total:

- 2 clear queries (single or multiple keywords, *e.g.*, *Markov chains*),
- 2 ambiguous queries (single or multiple keywords, *e.g.*, *architecture*),
- 2 metadata queries with the structure metadata:value (*e.g.*, *to:costache@L3S.de*, which translates to emails sent to "costache@L3S.de")

[22]For the evaluation of PIM systems, as the Desktop search tools, we are always faced with the problem of a very low number of testers —due to privacy concerns, people are not willing to provide their private data and participate in such experiments. However, we consider that for this kind of systems, the number of user who participated in the experiments (12) is quite reasonable.

Query type	P@1		P@2		P@3		P@4		P@5	
	B	B^{++}	B	B^{++}	B	B^{++}	B	B^{++}	B	B^{++}
clear	0.85	**0.91**	0.73	**0.89**	0.68	**0.89**	0.66	**0.89**	0.67	**0.87**
ambiguous	0.78	**0.82**	0.72	**0.84**	0.74	**0.83**	0.69	**0.84**	0.69	**0.83**

Table 2.3 P@1-5 for querying with Beagle and Beagle^{++}

- 2 queries of type metadata and some additional keywords (*e.g.*, *recommender to:costache@L3S.de*, which translates to querying for emails sent to "costache@ L3S.de" about "recommender" systems).

For the top-5 results, the user was asked to rate a query result with 0 (not relevant) or 1 (relevant). The user needed to consider a result only as relevant or not, disregarding the extent of the relevance. 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) Beagle^{++} using the same TFxIDF measure for ordering its output, but giving more importance to metadata results than to regular Desktop items[23], and (3) Beagle^{++} using enhancements for both metadata support and Desktop ranking based on ObjectRank.

We measured the quality of the produced annotations using *precision*, a standard IR evaluation measure. As the results had a confidence score, we computed precision at different levels, namely P@5, P@4, P@3, P@2, P@1. The precision at level K (P@K) is the precision score when only considering the Top-K output. It represents the number of relevant query results within the Top-K results divided by K, the total number of results considered. First, the P@K scores were computed for each user and query, then we averaged these values over the 2 queries of each type (clear and ambiguous), obtaining the user's opinion on each type of query. We further averaged over all subjects and excluded the outliers. We considered as outliers the results which were considerably distant from the average of the results, namely, not included within a range of 70% plus/minus from the average. This was done mainly because we observed that some users rated the relevance of some results incorrectly. The resulting values are listed in Tables 2.3 and 2.4, where B represents the results obtained using Beagle, B^{++} using Beagle^{++}, and B^{++}OR using Beagle^{++} enhanced with the ranking module.

Results and Analysis

In all cases, we observe that Beagle^{++} outperforms Beagle (see Table 2.3). This is in fact explainable, since Beagle only uses TFxIDF to rank its results, thus missing any kind of contextual importance measure for the Desktop resources. Another

[23]The advantage here is given by the fact of having metadata results additionally to normal Desktop items. Compared to the Beagle system, the ranking is enriched with additional metadata results.

Query type	P@1	P@2	P@3	P@4	P@5
	B^{++}	B^{++}	B^{++}	B^{++}	B^{++}
	B^{++}_{OR}	B^{++}_{OR}	B^{++}_{OR}	B^{++}_{OR}	B^{++}_{OR}
clear	0.86	0.84	0.84	0.84	0.82
	0.72	0.77	0.79	0.79	0.78
ambiguous	0.82	0.84	0.82	0.82	0.79
	0.82	**0.86**	**0.88**	**0.86**	**0.83**
metadata	0.92	0.85	0.83	0.83	0.83
	0.92	**0.88**	**0.89**	**0.88**	**0.87**
metadata & keyword	0.68	0.73	0.74	0.72	0.67
	0.77	**0.73**	0.70	0.65	0.63

Table 2.4 P@1-5 for querying with Beagle^{++} and Beagle^{++} with ObjectRank

observation is that, Beagle^{++} (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 (the ones she often uses), whereas the other automatically installed files (*e.g.*, help files, which she might never have used) are not associated with metadata, and thus ranked lower. When we have metadata describing a Desktop item, more and typically also very relevant text is available as part of the metadata to search for, and thus this item is also easier to find.

When comparing Beagle^{++} results against Beagle^{++} with ObjectRank (see Table 2.4), the second system proves to be weaker in the case of clear queries. This is because ObjectRank is mainly used for disambiguating and for a clear query it practically adds some noise in the results. The performance of the ObjectRank algorithm improves and gets better than TFxIDF scores in the case of ambiguous queries, since we are considering queries with at least two interpretations. In addition, it is natural for the algorithm to push at the top of the results' list the interpretation that is more used by the user, because most of the resources related to that sense would be interlinked and provide a better score.

In conclusion, the metadata enhancement solutions we proposed offer a visible quality improvement for Desktop search with any type of user query. Moreover, the Desktop ranking mechanism we introduced further enhances Desktop search output quality for ambiguous queries.

Benchmarking and System Performance

The second set of experiments we conducted aimed to compare the performance of the Beagle^{++} system against Beagle. All 12 data collections were indexed with both Beagle and Beagle^{++} and for each of the systems we observed the indexing time for

various dimensions of the data sets. On average, we had 4,707 resources per user, with an average size of 250,140 bytes per resource, which resulted in approximatively 1.17 GB of indexes. The range of Desktop resources to be indexed per user varied from as little as 213 to ca. 30,600. The average time of indexing with Beagle was 39.21 minutes, while with Beagle^{++} this increased to 149.24 minutes. This increase is quite normal, since besides the Beagle processes, our modules are also executed in Beagle^{++}, which takes more time. Also, the indexing in Beagle^{++} is performed in a completely different store than Beagle is using, the RDF repository which has a full-text index and also a semantic relations one (the querying is also performed on this store, and therefore the querying can also be slower). It is also important to note that this indexing is done only once, when Beagle^{++} is first run on a machine, and thus it does not impact the performance of our application. Afterwards, when a new action occurs (creation, deletion, moving, renaming of a file), only the particular affected file is handled, which is transparent to the user and makes use of very few resources. The obvious gain with the Beagle^{++} indexing is the additional metadata generated by the annotation tools — on average, Beagle^{++} generated about 1.8 million triples per user, useful data which is used for a better retrieval. An example showing the complexity of the generated Desktop metadata graphs is depicted in Figure 2.7.

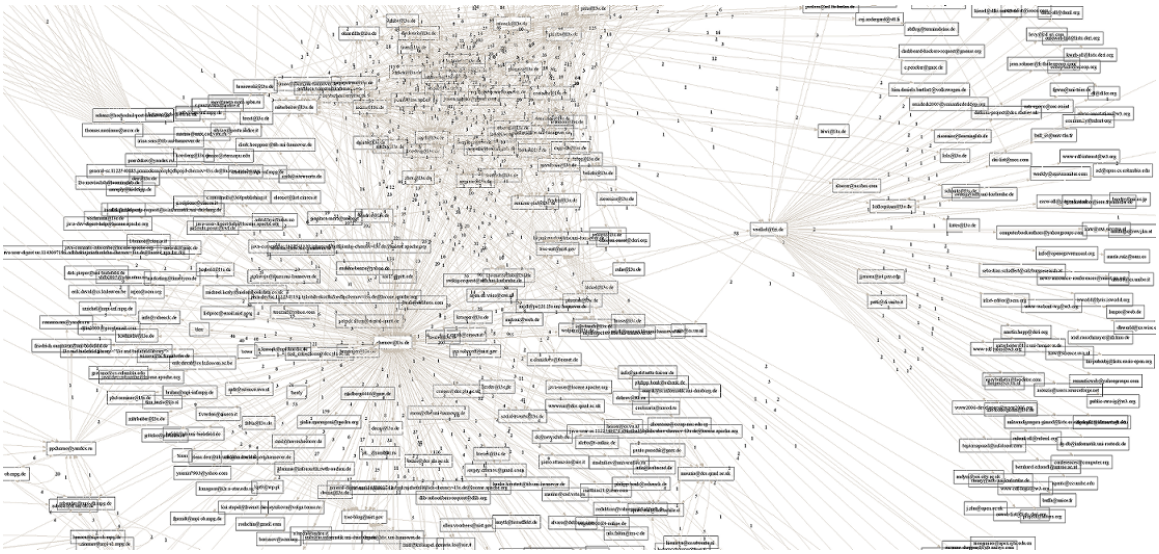

Figure 2.7 An example of the generated Desktop metadata graphs. Note that the image is fuzzy due to privacy concerns. Nevertheless, this excerpt visualises the connectivity of a Semantic Desktop's metadata.

For measuring the response time for a query, each user proposed a personal query and a total of 12 queries were run against each of the data collections. Three of the queries provided no result when Beagle was used as the search engine. For the same queries, Beagle^{++} provided more results in almost all cases. In Table 2.5 we show the number of results returned for one query in the case of all users. The

User	1	2	3	4	5	6	7	8	9	10	11	12
Beagle	0	0	2	2	1	2	8	1	5	8	14	4
Beagle^{++}	0	0	2	2	1	4	24	1	6	11	16	6

Table 2.5 Number of hits provided by Beagle and Beagle^{++} for one query

response time for these queries increased from an average of 0.348 seconds for Beagle, to 2.192 seconds for Beagle^{++} — still quite a good response time. The almost two seconds difference is due to Sesame and Lucene, both located at the NEPOMUK side: this takes longer than the Lucene index located in the Beagle system because of the overhead caused by the XML-RPC framework of NEPOMUK. Considering the number of found resources, Beagle^{++} outperforms Beagle with a significant increase from an average of 5.714 results per query, to 18.156 results, which means a bigger range of possible positive responses for the user to choose from (usually also presented in a better ranking order). Overall, the average response time for a query increased, from 0.372 seconds to 2.200.

2.3.6 Lessons Learned

Developing, evaluating, and using Beagle^{++} allowed us to realise several issues one needs to deal with in order to create an effective Semantic Desktop search engine. In this section we present and discuss the most important issues.

Metadata Extraction. Extracting metadata that are not explicitly stored in desktop resources requires non-trivial algorithms, or the usage of external information sources (*cf.* Metadata Filter for Scientific Publication in Section 2.3.2). For the former, we had to balance between performance and metadata quality, while considering their CPU and memory requirements. For the latter, a limited internet connectivity requires local copies of such information sources, which increases the application footprint. Those practical considerations are necessary to keep the application usable and accepted by users.

We further realised that when different filters use the same ontologies and URI schemes, *i.e.*, reusing unique identifiers from the resources being processed, data integrate nicely in the RDF repository. In case of the email metadata filter that reuses the message-id identifier, email and people will be connected to all relevant emails once they are stored in the RDF repository.

Integrating Extracted Metadata. Metadata that are not uniquely identifiable at filter-level need more sophisticated integration at the global-level in the RDF repository. The improvements we identified when using our approaches (see Section 2.3.3), allowed us to clearly realise that metadata enrichment methodologies are both feasible and necessary for providing effective search functionalities over metadata.

Using Metadata to Enrich Search Results. By extending the metadata so that users can search on and with structure, both the querying process as well as the search results have to handle that structure. On the one hand we had to balance between a simple query language and rich structured queries. On the other hand, the search result need to be visualised together with their structured metadata, without overwhelming the user with such information.

2.3.7 Discussion

In this section we presented the Beagle^{++} Desktop search tool and the underlying architectural design details for implementing a semantically enhanced Desktop search application. Our current implementation builds upon a snapshot of the standard Beagle implementation and we provided details about some new components we added to the system: the Metadata Filters and the Metadata Enrichment Components — ObjectRank and Attachment-File Linker.

Improvements are planned for every module, to offer a better performance in terms of efficiency in storage, indexing and querying. In order to improve the querying results, a task detection module is envisioned, which would be able to disambiguate the meaning of the user's query and push more valuable results to the user, as described next.

2.4 Desktop Context Detection Using Implicit Feedback

The personal information stored on the desktop usually reaches huge dimensions nowadays. Even providing additional metadata for resources, can sometimes not be sufficient to cope with the complex tasks that need to be fulfilled. Therefore, we consider that an efficient method of identifying the present working context would mean an easier management of the needed resources. Next we propose a new way of identifying desktop usage contexts, based upon a distance between documents, which also takes into account their access timestamps. We investigate and compare our technique with traditional term vector clustering, our initial experiments showing promising results with our proposed approach.

2.4.1 Context Detection on the Desktop

Looking at the previous work, text based clustering is a common method used for context detection. In this work we propose to also exploit usage analysis (i.e., access timestamps of documents) in the pursuit of this goal. This section will start with a description of the textual clustering methods we considered for desktop context

detection, then it will introduce a new activity based document similarity metric, and finally it will exploit this metric for grouping resources on the PC desktop.

Text Based Context Detection

Using term frequencies coordinates to construct a representational space leads to a good measure of similarity between documents, and we thus considered them as an appropriate input source for desktop context detection. We used the cosine distance to compare the term vectors representing two documents, and then clustered them using both K-means and an agglomerative algorithm with average-link and complete-link distances for the similarity between clusters (three approaches). We chose only these approaches (i.e. we rejected single-link), as they yield more dense clusters.

PC Desktops were defined as containing all data stored by a single user on a personal machine. This includes personal files (HTML, DOC, PDF, PPT, XML, etc.), web cache history, messenger history, emails, but also program generated indexable files (i.e., containing some form of textual content). Upon their full text, a standard text preprocessing technique was taken: tokenization, removal of stop words, and then stemming. Finally, the obtained word list needed to be further pruned, since it would have been computationally too expensive to apply clustering on a term vector space with so many dimensions, the word list reaching almost 300,000 stems. This is also partially due to the fact that the text extractors generated a lot of noisy words (e.g., by sticking some words together). We thus followed the approach of Yang and Pedersen [YP97], and pruned our word list by document frequencies, keeping only the words with the highest number of occurrences, as they were proved to be better for aggressive dimension reduction than those with low document frequencies. We used 15 as the minimum DF for the pruning threshold, which reduced the list of words to around 1,000 entries. The newly obtained list was used to compute the TF vectors for each document, the input for the clustering algorithms. The YALE[24] system provided us with the functionalities and algorithms implementations needed to perform the clustering. The results of these methods will be presented in Section 2.4.2.

Activity Based Context Detection

In order to apply a standard clustering algorithm exploiting implicit feedback, we first need to define a distance between documents which exploits this information. The main idea of our activity-based distance is that if two files are *often* accessed in a *small window of time*, the distance between them should be small. One obvious parameter of the distance is then the time $t(a_f, b_g) = |t(a_f) - t(b_g)|$, elapsed between two file accesses a_f and b_g to files f and g respectively. We also consider an additional parameter, namely the number of steps $s(a, b)$ between accesses a and b. If we note s_x

[24] http://yale.cs.uni-dortmund.de/

the position of access x in the file access sequence, then $s(a_f, b_g) = |s(a_f) - s(b_g)|$. For example, if we consider the sequence of file accesses $a_f \rightarrow c_h \rightarrow b_g$, then $s(a_f, b_g) = 2$. We therefore propose an activity-based distance between file accesses, and define it as following product:

$$d(a_f, b_g) = t(a_f, b_g)s(a_f, b_g) \tag{2.2}$$

Before exploiting this file access distance to measure distances between files, let us first observe that if two files f and g are accessed one time, separated by a given number of steps, they should have a distance lower than if they were accessed a larger number of times separated by the same number of steps. Furthermore, a file can obviously be accessed more than one time. For example, let us consider two distinct accesses a_f and b_f to the same file f. If we use Equation 2.2 as a distance between files, then $d(f, f) = d(a_f, b_f) = t(a_f, b_f)s(a_f, b_f) > 0$, which is therefore not a valid distance. To avoid this, we introduce a new function $d'_i(f, g, \sigma)$, defined as follows: Consider the log of all file accesses. Let us assume, without loss of generality, that among f and g, f appears first in the log. Also, let us denote a^i_f as the i^{th} access to file f if $i \leq n_f$, or $a^i_f = N$ if $i > n_f$; where n_f is the total number of accesses to file f, and N is the total number of access logged. We can now define:

$$d'_i(f, g, \sigma) = \begin{cases} d(a^i_f, a_g) & \text{if } \exists\ a_g | s(a^i_f) \leq s(a_g) < s(a^{i+1}_f) \\ & \wedge\ s(a^i_f, a_g) = \sigma \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

After defining the function $occ(f, g, \sigma)$ as the number of times file g is accessed σ steps after f, we can finally derive our activity based distance, which is the sum of the closest file access distances weighted by the inverse of their occurrences:

$$D(f, g) = \sum_{\sigma} \left[occ(f, g, \sigma)^{-1} \sum_{i} d'_i(f, g, \sigma) \right] \tag{2.4}$$

Having this distance in place, we applied an agglomerative clustering algorithm with both complete- and average-link. We could then cut the tree at different heights in order to obtain any number of non overlapping clusters representing contexts.

2.4.2 Experiments

We evaluated how the newly introduced activity based context detection methods perform when compared to text based one. The experiments were performed only on a small scale within the research environment of L3S. All file accesses (open, create, etc.) on the desktop were logged during several months of normal activity, with an installed activity logging tool[25]. In the end, the log files were used to cluster their resources based on our new methods.

[25]We thank Leo Sauermann from DFKI for implementing this logger.

We experimented on several values of the number of clusters that were supposed to exist on the desktop. In order to make sure the resulted contexts are humanly apprehensible, we fixed the average size of a cluster at 20 documents, which yielded a fixed number of clusters. As only a low percentage of the total number of files on the desktop are actually touched by the user, the activity based methods yielded a lot less clusters. Therefore, in order to increase the quality of our experiments, we also experimented with limiting the text clustering only to the user accessed files. To summarize, we used the following methods:

1. Text clustering over all documents
2. Text clustering over touched documents
3. Activity clustering over touched documents

We first analyze the text clustering performed on the whole set of documents on user's desktop, namely Method 1 as presented above. When manually inspecting the obtained clusters, we could easily see that the agglomerative clustering output one or two huge clusters and the rest were grouped in smaller ones (1-5 documents), which would not be the expected outcome, a relatively uniform distribution of resources in clusters. The K-means version, produced a slightly better repartition among clusters. In the larger clusters however, we could observe that there were several clearly outlined sub-clusters that were grouped together. We believe this to be the consequence of imposing the number of clusters (to K), and thus better results might be obtained after a better definition of K.

The activity clustering method performed almost the same as K-means applied with Method 1, meaning that the documents in the test set were distributed more equally among the clusters than text clustering, and the larger clusters also had one or two sub-clusters inside. If we compare Methods 2 and 3, clustering on the set of the touched documents, we could observe an encouraging behavior of both clustering methods. They managed to cluster files that normally were not so easy to connect, such as papers with the presentations about them, etc.

The activity clustering is very promising, as it does not receive any kind of information about the content of the files, as the text clustering. More, the activity clustering brings several advantages, as it filters out uninteresting documents from the user's work space, while also reducing the dataset considerably. Finally, combinations of these two sources of information will probably result in significant improvements over each of them taken alone.

2.4.3 Discussion

In the previous sections we proposed to use usage analysis as an input source for clustering desktop documents. We defined a distance between documents, taking into account the number of steps between consecutive accesses of files and a time window in which they occur, and applied to it agglomerative with complete and average link

clustering methods. Our initial experiments showed promising results, our proposed usage analysis based approach performing similarly to textual clustering, even though it has *no* information about the textual content of documents.

As future work, we plan to derive time coordinates for the files based on their activity distances, allowing us to apply clustering algorithms necessitating also coordinates (e.g. K-means). We also intend to apply more complex clustering algorithms, such as a fuzzy K-means approach, which would allow for one file to belong to different clusters in a certain proportion. Finally, combining text and activity based clustering would also be an interesting idea to investigate, as well as performing a more semantic-based clustering (e.g., based on the locations of these files on the PC Desktop).

2.5 Desktop Context Detection Using Bayesian Networks

As already argued before, we believe that a good understanding of a user's (working) contexts provides the basis for improved desktop information management as well as for personalized desktop and Web search. We propose to combine a variety of evidences found by analyzing desktop information for inferring the user's working contexts or more precisely file-to-context assignment using a Bayesian network. Our preliminary experiments focus on identifying a good selection of evidences to use and show that the choice of evidences is coherent with user assessments for desktop files, as well as the contexts inferred by the Bayesian network.

2.5.1 Context Detection Evidences

As presented in Section 2.2, many approaches focus on identifying user desktop behavior using evidences gathered from the desktop. Our evidences are mainly extracted from user actions and summarized within behavioral patterns, which can then be used to model user contexts. We now present the different desktop evidences we collect and explain their transformation into similarity measures which we later use to construct the BN.

Textual Properties (T). Many approaches consider grouping together files (e.g., clustering) based solely on their textual properties. We follow [RMO+93], which focuses on desktop resources, and represent each text document as a vector of TFxIDF coordinates. Cosine similarity (or angle distance) between such vectors is used to model the similarity of desktop resources. When the vector similarity is small, we consider the documents similar, as typically done in Information Retrieval.

Usage Analysis (UA). Usage analysis refers to information when a file is active for a user, along with the file access times. The collected access times are used

to compute the distance between any two resources using the sequence and session activity similarities described below. Our approach is based on [CCGN06, GCC+08], which showed that grouping desktop resources and ranking based on usage analysis is quite beneficial. We present hereafter a reworked version of the so-called activity distance as a similarity.

We need to consider different factors when defining an activity similarity between two desktop resources. More specifically, two resources are more similar: (i) if they are mostly used in a small interval of *time*, (ii) if their access times are nearer in an *access sequence*, and (iii) if they have many *occurrences* of close accesses in time or sequence.

According to the above we represent accesses to a desktop resource as two signals: (1) a continuous signal along time, and (2) a discrete signal along the sequence dimension. Let us denote the time signal of resource r as $f_r(t)$ and its sequence signal as $g_r(s)$, where t is the time and s is the steps where r is accessed in the access sequence. Each time r is accessed, we add to f a curve following a normal distribution centered on the time t_i at which the resource is accessed. Similarly, for the sequence signal, we add a curve following a binomial distribution – which is the discrete equivalent to a normal distribution – centered on the step s_i at which the resource is accessed. If we note T the list of all times where r is accessed, and S the list of all sequence steps where r is accessed, then the signals for r are:

$$f_r(t) = \sum_{t_i \in T} N_{t_i,\sigma}(t) = \sum_{t_i \in T} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-t_i)^2}{2\sigma^2}}, and \tag{2.5}$$

$$g_r(s) = \sum_{s_i \in S} B_p(s - s_i) = \sum_{s_i \in S} \binom{N}{s - s_i} p^{s-s_i}(1-p)^{N-(s-s_i)}$$

Thus, the more similar the signals of two resources are, the more similar the resources themselves are to each other. We can therefore use the correlation coefficient of the signals of two resources to measure their similarity. The activity session similarity between two resources a and b is then:

$$winCC(f_a(t), f_b(t)) = \frac{cov(f_a(t), f_b(t))}{std(f_a(t)) \cdot std(f_b(t))}, \tag{2.6}$$

and the sequence similarity is:

$$seqCC(g_a(t), g_b(t)) = \frac{cov(g_a(t), g_b(t))}{std(g_a(t)) \cdot std(g_b(t))}, \tag{2.7}$$

where *cov* is the covariance and *std* the standard deviation of the two signals. Since the correlation coefficients are bounded in the interval $[-1; 1]$, we define the *usage analysis* similarity as:

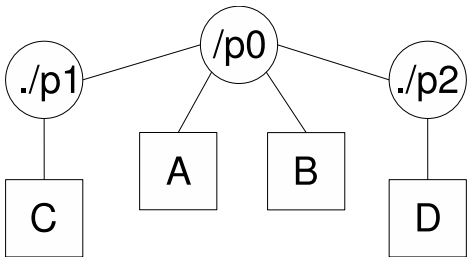

Figure 2.8 Hierarchy example (circles are directories and squares files).

$$ua(a,b) = \frac{winCC(f_a(t), f_b(t)) + 1}{2} \cdot \frac{seqCC(g_a(t), g_b(t)) + 1}{2},$$

and it is bounded in the interval $[0, 1]$.

Files Opened Concurrently (FOC). According to related studies (Section 2.2), when several resources are accessed in parallel at the same time they are (to some extent) related. UA logs the switching between resources, and FOC just considers the resources simultaneously accessed (resources displayed by a window at a given time), information not necessarily captured by UA.

In this case we propose to provide the probability that two resources belong to the same context. Let O_a be the number of times a is accessed alone, and O_b when b is accessed alone. Let also O_{ab} be the number of times a and b are accessed concurrently. We define the FOC similarity between resources a and b as $foc(a,b) = \frac{O_{ab}}{O_a+O_{ab}+O_b}$.

Folder Hierarchy (FH). Directories allow to classify files, thus enabling the user to later retrieve them by browsing. It is therefore reasonable to consider files residing within the same directory have a higher probability of being related. This corresponds to an explicit user defined context.

Note that files sharing a common path prefix are also, in a restricted sense, in the same folder, i.e. they have a common prefix. The fact that some files are in subdirectories of the common directory indicates however a lower probability to belong to the same context.

Intuitively, files A and B in Figure 2.8 have the highest probability to belong to the same context. Files A and C are a bit less probable to belong to the same context, since they don't have exactly the same path: they are both under $/p0$ but C is also under $./p1$. And files C and D are less probable to belong to the same context since they are in two different subcontexts (i.e., subdirectories) of $/p0$.

Considering the above we propose to use the shortest path between two files in the file system, modeled as a tree with directories as branches and files as leaves. In our example, files C and D would have a distance of 2. Since in our approach we need a similarity and not a distance we define the FH similarity as $fh(a,b) = 1/(1 + shortestPath(a,b))$.

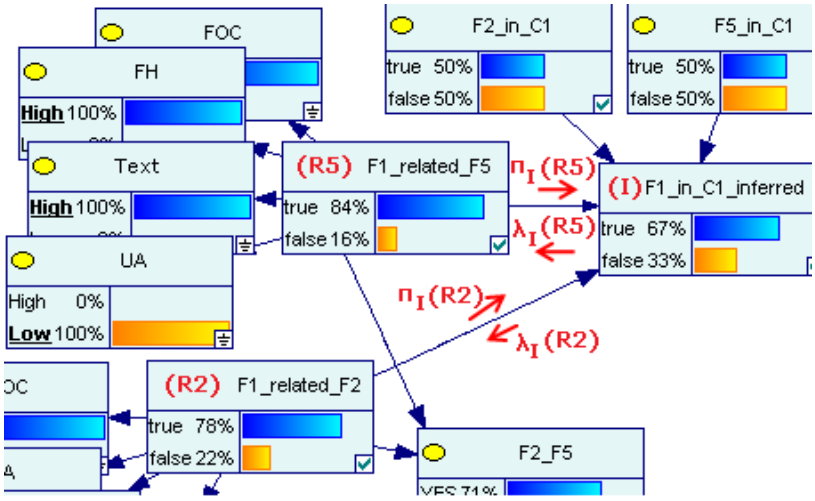

Figure 2.9 Small part of an example BN.

2.5.2 The Context Bayesian Network

We build a BN based on the evidences gathered from the user activities on the desktop, but also on direct input from the user, concretized into an initial user assessment of 100 random files from the desktop which the user had to classify into several contexts. The respective contexts will then be filled with other related files – as inferred by the BN. In addition, related files outside the identified contexts will be detected, which are a source for new contexts.

The BN is constructed from the following nodes:

- **Evidence Nodes** describe the type of evidences we have for each particular file. They can be text, UA, FOC, or FOH, as described in Section 2.5.1. These particular nodes will just describe that two files are related and with what evidence type.

- **Directly Related Nodes** describe the relationship between two files, based on Evidence Nodes. It is practically an unification of evidences that can be gathered for two files. Therefore, direct links between a Directly Related Node and the respective Evidence Nodes exist (see node "F1_related_F5" in Figure 2.9).

- **Inferred Related Nodes** are similar as significance as the previous type of nodes, since, they also express a possible relationship between files. However, this type of nodes is suited for pairs of files for which we don't have direct evidence (text, UA, FOC, or FH), but their relationship can be transitively inferred from the Directly Related Nodes. For example, if we have two Directly Related Nodes which express the relationship between (F1, F2) and (F1, F5) (see Figure 2.9), we infer relationship (F2, F5) in this type of node.

- **Context Nodes** are constructed from the direct feedback given by the user for the randomly selected files. They simply show that a file belongs to a context.

- **Inferred Context Nodes** also rely on a transitive relationship inferred when

we know there is a relationship (direct or inferred) between two files – (F1, F5), and we also know that one file belongs to a context – (F5, C1). Then we can easily infer that F1 should also belong to the same context – (F1, C1) (see Figure 2.9).

Construction of the BN. In order to construct the BN, we first collect the evidences that we have for the similarity between files, computed by comparing each two files on the desktop. Whenever an evidence similarity (text, UA, FOC, FH) is above a given threshold, we construct an Evidence Node. Probability tables for these nodes are deducted from user questionnaires. In these questionnaires, users had to evaluate how much would such a single evidence influence the similarity of two files, each evidence by itself.

At the same time, we also add the Directly Related Nodes, which will be connected to the appropriate Evidence Nodes, as described above. For example, if we have significant (i.e., higher than a threshold) text and FH evidences that F1 is related to F2, we construct the two Evidence Nodes for text and FH evidences, but we also construct a Directly Related Node, which means that F1 and F2 are related, and we also link this node to the two Evidence Nodes for text and FH.

Once all these nodes are added, we construct the Inferred Related Nodes – if the Direct Related Nodes referring to F1 related to F2, and F2 related to F3 exist, but the Direct Related Node implying that F1 is related to F3 does not exist, we then construct an Inferred Related Node expressing this relationship between F1 and F3, which will link to the first two existing Direct Related Nodes.

Next, we construct new nodes based on the user evaluation on the randomly chosen files from her desktop. The user had to name contexts and also assign files to them. She could also express her distrust, like putting a 0 (in a table) if the respective file is not in a specific context. But, we only took into account the positive evidences and constructed Context Nodes. Also, these nodes needed to be linked to the nodes that express there is a relationship between two files - the Directly Related Nodes and also the Inferred Related Nodes, in order to be able to make new inferences: if F1 is related (directly of inferred) to F2 and if F1 is in context C, then we should infer that also F2 has to be in context C. Of course, some restrictions need to be applied: we also check if there exists negative evidence from the user that F2 should not belong to context C, or if the Context Node F2 in context C was not already generated from the evidences from the user.

Inference & Context-Files Identification. Once the construction of the BN is completed, we determine the probability of cause nodes (file to context assignment) given the probability of the observed effect (evidences), and identify which files belong to each context. This task is performed using Pearl's probabilistic inference (PI) [Pea88]. This algorithm is interactive and in each step one node is activated for calculating its belief. The activated node N recomputes its own belief using messages collected from its parents ($\pi_N(P_i)$) and its children ($\lambda_N(C_j)$). Once the node has its belief, it sends messages to its parents ($\lambda_N(P_i)$) and its children ($\pi_{C_j}(N)$), which are then used when these nodes are activated.

Consider the small BN example in Figure 2.9. Node I computes its belief using messages from its parents and children, which include $\pi_I(R2)$ and $\pi_I(R5)$. Once the belief is computed, node I sends new messages to its parents and children, including messages $\lambda_I(R2)$ and $\lambda_I(R5)$. As such, node $R2$ is affected by the belief of node $R5$ despite the fact that they are not directly connected.

Once PI finishes, the Inferred Context Nodes provide probabilistic information about each file belonging to a context. Also, some other type of information is being offered to the user, the one generated by the Inferred Related Nodes for files about which nothing could have been inferred relative to their belonging to a context. This allows us to identify new contexts which the user did not specifically name in her initial input.

2.5.3 Experiments

Experimental Setup. We define a user desktop as containing all data stored by a user on a personal machine. This includes personal files (e.g., HTML, DOC, PDF, PPT, XML, JPG), Web cache history, messenger history, emails. Upon their full text, standard text preprocessing techniques were performed: tokenization, stop words removal, stemming. We then computed the similarities for each pair of desktop resources, as described in Section 2.5.1.

We evaluated how the newly introduced activity based context detection methods perform. The experiments were performed only on a small scale (i.e., the authors of this paper). All file accesses (open, create, etc.) on the desktop were logged during several months of normal activity, with an installed activity logging tool. The log files were used to provide the additional similarity measures used.

Experimental Results and Comments

realScore	0.33	0.5	0.66	0.75	1
from total number of pairs	100%	92.08%	90.67%	89.89%	89.89%

Table 2.6 True positive evidences.

True positive evidences. For this type of evaluation we want to see if for two files that the user assessed as being in the same context (i.e., related), the evidences that we generate also support this fact – their values are higher than chosen thresholds, and therefore considered positive evidences. For each two such files, we consider the fraction of positive evidences from the total of evidences for a pair of files as the realScore. Then, we count the percent of the pairs of files from all files for which the realScore is higher or equal to 1, 0.75, 0.66, 0.5, 0.33 : 89.89%, 89.89%, 90.67%, 92.08%, 100%, as shown in Table 2.6. These first results show that the generated

evidences are correct, since at least 33% of the generated evidences for related files are shown to be positive, and 89.89% of the pairs of related files have all evidences positive.

True negative evidences. This measure shows that the negative evidences (lower than a threshold) about two files are in correlation with the fact that the user specifically said that two files are not related – one file is in one context and the other is not in the same context. We compute for all such pairs, the number of negative evidences, and divide it to the total number of evidences for the same pair of files and get again a realScore. Then, we compute the percentage of the pairs for which the realScore is higher or equal to 1, 0.75, 0.66, 0.5, 0.33: 76.87%, 76.87%, 99.06%, 99.49%, 100%, as shown in Table 2.7. This again supports the type of evidences that we generated, saying that at least 33% of the generated evidences for unrelated files are negative, and 76.87% of the pairs of unrelated files have all evidences negative.

realScore	0.33	0.5	0.66	0.75	1
from total number of pairs	100%	99.49%	99.06%	76.87%	76.87%

Table 2.7 True negative evidences.

BN Performance. The user assessment was split into three parts, and two thirds were used for training the BN and one third used for evaluation. Then, the evaluation set was varied from within the three chunks of the user's assessment. For each of the three iterations, precision and recall were computed for each of the user clusters, and then averaged over all clusters. The preliminary results are supporting our ideas – precision was of 77.78% and recall 73.97%. However, further experiments are required to validate these results in a higher variety of settings: an increased number of users, various threshold values for choosing the positive evidences, as well as different initial probability table assignments in the BN, or an increased number of assessed files.

2.5.4 Discussion

In the previous sections we presented innovative similarity measures used for detecting user contexts through a BN. Our preliminary experiments show promising results regarding the choice of evidences with respect to user assessments (both positive and negative), and also good results regarding the output of the BN and its capability of inferring user contexts. In future work we plan to extend these experiments, by varying different parameters which influence our results, in order to confirm them on a wider scale.

2.6 P-TAG: Large Scale Automatic Generation of Personalized Annotation TAGs for the Web

Free form metadata or *tags*, as used in social bookmarking and folksonomies based systems, have become more and more popular and successful. Such tags are relevant keywords associated with or assigned to a piece of information (e.g., a Web page), thus describing the item and enabling keyword-based classification. In this work we propose P-TAG, a method which automatically generates personalized tags for Web pages. With current applications which tend to construct the personal desktop not only from resources stored on the PC, but to expand it on the web, the personal annotations can bring homogeneity within the personal resources. Keywords are generated based on the content of the Web page but also based on the content of the user's Desktop, thus expressing a personalized viewpoint very relevant for personal tags. We implemented and tested several algorithms for this approach and evaluated the relevance of the resulting keywords. These evaluations showed very promising results and we are therefore very confident that such a user oriented automatic tagging approach can provide large scale personalized metadata annotation as an important step towards expanding the personal Desktop on the Web.

2.6.1 Automatic Personalized Web Annotations

We distinguish three broad approaches to generate personalized Web page annotations, based on the way user profiles (i.e., Desktops in our case) are exploited to achieve personalization: (1) a document oriented approach, (2) a keyword oriented approach, and (3) a hybrid approach.

Document Oriented Extraction

The general idea of the document oriented approach is as follows. For a given Web page, the system retrieves similar documents from the personal Desktop. This set of related personal documents reflects user's viewpoint regarding the content of the Web page. Hence, the set will be used to extract the relevant annotations tags for the input Web page. The generic technique is also depicted in the algorithm below.

Algorithm 2.6.1. Document oriented Web annotations generation.

1: Given a browsed Web page p,
2: **Find** similar Desktop documents $DS_i, i \in \{1, \ldots, nd\}$
3: **For** each document DS_i:
4: **Extract** its relevant keywords.
5: **Select** Top-K keywords using their confidence scores.

We will now detail the specific algorithms for accomplishing steps 2 and 4 in the following two subsections.

Finding Similar Documents

We consider two approaches for this task, namely Cosine Similarity (i.e., nearest neighbor based search), and Latent Semantic Analysis (i.e., clustering based search). In both cases, we apply the algorithms only to those terms with a Desktop document frequency above 10 and below 20% from all Desktop documents of the current user. This is necessary in order to avoid noisy results (i.e., documents with many words in common with an input page p, yet most of these being either very general terms, or terms not describing the interests of the surfer). Moreover, such optimizations significantly improve computation time.

Cosine Similarity. Nearest neighbor search, based on TFxIDF (Term Frequency multiplied by Inverse Document Frequency), outputs a similarity score between two documents utilizing the following simple formula:

$$Sim(D_i, D_j) = v(D_i) \cdot v(D_j) = \sum_{t \in D_i \cup D_j} w_{i,t} \cdot w_{j,t} \tag{2.8}$$

where $v(D_i), v(D_j)$ are the term vectors describing documents D_i, D_j using TFxIDF, as within the Vector Space Model [BYRN99], and $w_{k,t}$ represents the actual TFxIDF weight associated to term t within document D_k.

Latent Semantic Analysis. Similar to clustering, LSA can compute document to document similarity scores. However, we decided to abandon experimenting with this algorithm, as it turned out to be too computationally expensive for regular Desktops consisting of several dozens of thousands of indexable items. Nevertheless, since the quality of its results might be high, we are currently investigating several techniques to optimize it or to approximate its results in order to enable evaluating it in a further work.

Extracting Keywords from Documents

Introduction. Keyword extraction algorithms usually take a text document as input and then return a list of keywords, which have been identified by exploiting various text mining approaches. To ease further processing, each keyword has associated a value representing the confidence with which it is thought to be relevant for the input document. Once such keywords have been identified for the input set of relevant Desktop documents (i.e., as determined using the previously described techniques), we propose to generate annotations for the original input Web page by sorting all these generated terms utilizing their confidence levels, and then taking the Top-K of them.

We investigated three broad approaches to keyword extraction with increasing levels of granularities, denoted as "Term and Document Frequency" (keyword oriented), "Lexical Compounds" (expression oriented) and "Sentence Selection" (summary oriented).

Term and Document Frequency. As the simplest possible measures, TF and DF have the advantage of being very fast to compute. Moreover, previous experiments showed them to yield very good keyword identification results [BH99, Eft95]. We thus associate a score with each term, based on the two statistics (independently). The TF based one is obtained by multiplying the actual frequency of a term with a position score descending as the term first appears more towards the end of the document. This is necessary especially for longer documents, because more informative terms tend to appear towards their beginning [BH99]. The complete TF based keyword extraction formula is as follows:

$$TermScore = \left[\frac{1}{2} + \frac{1}{2} \cdot \frac{nrWords - pos}{nrWords}\right] \cdot TF \tag{2.9}$$

where $nrWords$ is the total number of terms in the document, pos is the position of the first appearance of the term, and TF is the frequency of each term in the considered Desktop document.

The identification of suitable expansion terms is even simpler when using DF: The terms from the set of Top-K relevant Desktop documents is ordered according to their DF scores. Any ties are resolved using the above mentioned TF scores [Eft95].

Note that a hybrid TFxIDF approach is not necessarily efficient, since one Desktop term might have a high DF on the Desktop, yet it may occur rarely on the Web. For example, the term "RDF" could occur frequently accross the Desktop of a Semantic Web scientist, thus achieving a low score with TFxIDF. However, as it is encountered more rarely on the Web, it would make a better annotation than some generic keyword.

Lexical Compounds. Anick and Tipirneni [AT99] defined the *lexical dispersion hypothesis*, according to which an expression's lexical dispersion (i.e., the number of different compounds it appears in within a document or group of documents) can be used to automatically identify key concepts over the input document set. Although there exist quite several possible compound expressions, it has been shown that simple approaches based on noun analysis produce results almost as good as highly complex part-of-speech pattern identification algorithms. We thus inspect the selected Desktop documents (i.e., at step 1 of the generic algorithm) for all their lexical compounds of the following form:

$$\{\ adjective?\ noun+\ \}$$

We note that all such compounds could be easily generated off-line, at Desktop indexing time, for all the documents in the local repository. Moreover, once identified, they could be further sorted depending on their dispersion within each document in order to facilitate fast retrieval of the most frequent compounds at run-time.

Sentence Selection. This technique builds upon sentence oriented document summarization: Having as input the set of Desktop documents highly similar to the input Web page, a summary containing their most important sentences is generated as output. Thus, sentence selection is the most comprehensive of these approaches,

as it produces the most detailed annotations (i.e., sentences). Its downside is that, unlike the first two algorithms, its output cannot be stored efficiently, and thus it cannot be precomputed off-line. We generate sentence based summaries by ranking the document sentences according to their salience score, as follows [LAJ01]:

$$SentenceScore = \frac{SW^2}{TW} + PS \ \ [+\frac{TQ^2}{NQ}]$$

The first term is the ratio between the square amount of significant words within the sentence and the total number of words therein. A word is significant in a document if its frequency is above a threshold as follows:

$$TF > ms = \begin{cases} 7 - 0.1 * [25 - NS] & , if\ NS < 25 \\ 7 & , if\ NS \in [25, 40] \\ 7 + 0.1 * [NS - 40] & , if\ NS > 40 \end{cases}$$

with NS being the total number of sentences in the document (see [LAJ01] for more details). The second term is a position score. We set it to $1/NS$ for the first ten sentences, and to 0 otherwise. This way, short documents such as emails are not affected, which is correct, since they usually do not contain a summary in the very beginning. However, it is known that longer documents usually do include overall descriptive sentences in the beginning [Edm69], and these sentences are thus more likely to be relevant. The final term is an optional parameter which is not used for this method, but only for the hybrid extraction approaches (see Section 2.6.1). It biases the summary towards some given set of terms, which in our case will correspond to the terms extracted from the input Web page. More specifically, it represents the ratio between the square number of set terms present in the sentence and the total number of terms from the set. It is based on the belief that the more terms in the set are contained in a sentence, the more likely will that sentence convey information highly related to the set, and consequently to the Web page to annotate.

Keyword Oriented Extraction

In the keyword oriented approach we start from extracting some keywords from the Web page to annotate. This set of keywords reflects a generic viewpoint on the document. In order to personalize this viewpoint, we then align these keywords with related terms from the personal Desktop, as in the algorithm presented below.

Algorithm 2.6.2. Keyword oriented Web annotations generation.

1: Given a browsed Web page p,
2: **Extract** relevant keywords from p
3: **For** each relevant keyword
4: **Find** related keywords on the local Desktop.
5: **Select** Top-K keywords using their confidence scores.

Step 1 can be accomplished using the algorithms from Section 2.6.1 - Extracting Keywords from Documents. The next two subsections will introduce the techniques we propose for the keyword alignment process, i.e., the finding of similar keywords on the Desktop corpus, namely (1) term co-occurrence statistics and (2) thesaurus based extraction.

Term Co-occurrence Statistics. For each term, one could easily compute off-line those terms co-occurring with it most frequently in a collection, such as user's Desktop, and then exploit this information at run-time in order to infer keywords highly correlated with the content of a given Web page. Our generic Desktop level co-occurrence based keyword similarity search algorithm is as follows:

Algorithm 2.6.3. Co-occurrence based keyword similarity search.

Off-line computation:
1: **Filter** potential keywords k with $DF \in [10, \ldots, 20\% \cdot N]$
2: **For** each keyword k_i
3: **For** each keyword k_j
4: Compute SC_{k_i,k_j}, the similarity coefficient of (k_i, k_j)

On-line computation:
1: **Let** S be the set of terms potentially similar to E,
the set of keywords extracted from the input Web page.
2: **For** each keyword k of E:
3: $S \leftarrow S \cup TSC(k)$, where $TSC(k)$ contains the Top-K terms most similar to k
4: **For** each term t of S:
5a: **Let** $Score(t) \leftarrow \prod_{k \in E}(0.01 + SC_{t,k})$
5b: **Let** $Score(t) \leftarrow \#Hits(E|t)$
4: **Select** Top-K terms of S with the highest scores.

The off-line computation needs an initial trimming phase (step 1) for optimization purposes. Also, once the co-occurrence levels are in place and the terms most correlated with the keywords extracted from the input Web page have been identified,

one more operation is necessary, namely calculating the correlation of each proposed term with the entire set of extracted keywords (i.e., with E), rather than with each keyword individually. Two approaches are possible: (1) using a product of the correlation between the term and all keywords in E (step 5a), or (2) simply counting the number of documents in which the proposed term co-occurs with the entire set of extracted keywords (step 5b). Small scale tuning experiments performed before the actual empirical analysis indicated the latter approach to yield a slightly better outcome. Finally, we considered the following Similarity Coefficients [KC99]:

- *Cosine Similarity*, defined as:

$$CS = \frac{DF_{x,y}}{\sqrt{DF_x \cdot DF_y}} \tag{2.10}$$

- *Mutual Information*, defined as:

$$MI = log \frac{N \cdot DF_{x,y}}{DF_x \cdot DF_y} \tag{2.11}$$

- *Likelihood Ratio*, defined below.

DF_x is the Document Frequency of term x, and $DF_{x,y}$ is the number of documents containing both x and y.

Dunning's Likelihood Ratio λ [Dun93] is a co-occurrence based metric similar to χ^2. It starts from attempting to reject the null hypothesis, according to which two terms A and B would appear in text independently from each other. This means that $P(A|B) = P(A|\neg B) = P(A)$, where $P(A|\neg B)$ is the probability that term A is *not* followed by term B. Consequently, the test for independence of A and B can be performed by looking if the distribution of A given that B is present is the same as the distribution of A given that B is not present. Of course, in reality we know these terms are not independent in text, and we only use the statistical metrics to highlight terms which are frequently appearing together. We thus compare the two binomial processes by using likelihood ratios of their associated hypotheses, as follows:

$$\lambda = \frac{max_{\omega \in \Omega_0} H(\omega; k)}{max_{\omega \in \Omega} H(\omega; k)} \tag{2.12}$$

where ω is a point in the parameter space Ω, Ω_0 is the particular hypothesis being tested, and k is a point in the space of observations K. More details can be found in [Dun93].

Thesaurus Based Extraction. Large scale thesauri encapsulate global knowledge about term relationships. Thus, after having extracted the set of keywords describing the input Web page (i.e., as with Step 1 of Algorithm 2.6.2), we generate an additional set containing all their related terms, as identified using an external thesauri. Then, to achieve personalization, we trim this latter set by keeping only those terms that are frequently co-occurring over user's Desktop with the initially identified keywords. The algorithm is as follows:

Algorithm 2.6.4. Thesaurus based keyword similarity search.

1: **For** each keyword k of a set P, describing Web page p:
2: **Select** the following sets of related terms using WordNet:
2a: Syn: All Synonyms
2b: Sub: All sub-concepts residing one level below k
2c: Super: All super-concepts residing one level above k
3: **For** each set S_i of the above mentioned sets:
4: **For** each term t of S_i:
5: **Search** the Desktop with $(P|t)$, i.e., the original set, as expanded with t
6: **Let** H be the number of hits of the above search (i.e., the co-occurrence level of t with P)
7: **Return** Top-K terms as ordered by their H values.

We observe three types of term relationships (steps 2a-2c): (1) synonyms, (2) sub-concepts, namely hyponymes (i.e., sub-classes) and meronymes (i.e., sub-parts), and (3) super-concepts, namely hypernymes (i.e., super-classes) and holonymes (i.e., super-parts). As they represent quite different types of association, we investigated them separately. Furthermore, we limited the output expansion set (step 7) to contain only terms appearing at least T times on the Desktop, in order to avoid any noisy annotations, with $T = \min(\frac{N}{\text{DocsPerTopic}}, \text{MinDocs})$. We set $DocsPerTopic = 2,500$, and $MinDocs = 5$, the latter one coping with the case of having small Desktops.

Hybrid Extraction

The hybrid approach mixes the document oriented approach with the keyword oriented one. Thus, the system first extracts some relevant keywords from an input Web page. This set of keywords is the same as in Step 1 of Algorithm 2.6.2 and reflects a generic viewpoint on the document. We personalize this viewpoint by retrieving the Desktop documents most relevant to it (i.e., using Lucene Desktop search) and then by extracting relevant keywords from them. The generic algorithm is as follows:

Algorithm 2.6.5. Hybrid generation of Web annotations.

1: Given a browsed Web page p,
2: **Extract** the relevant keywords P_i from p
3: **For** each keyword P_i:
4: **Find** the most relevant Desktop documents $D_{i,j}$.
5: **For** each document $D_{i,j}$:
6: **Extract** its relevant keywords.
7: **Select** Top-K keywords using their confidence levels.

Steps 2 and 6 utilize the keyword extraction techniques described in Section 2.6.1 - Extracting Keywords from Documents. Step 4 represents a regular Desktop search, in which the results are ordered by their relevance to the query, i.e., to each keyword P_i.

2.6.2 Experiments

Experimental Setup

System Setup. We have asked 16 users (PhD and Post-Doc students in various areas of computer science and education) to support us with the experiments. Our Lucene[26] based system was running on each personal Desktop. Lucene suited our interests best, given its rapid search algorithms, its flexibility and adaptivity, and last but not least its cross-platform portability. Thus, each user's corpus of personal documents has been indexed for later faster retrieval. More specifically, we indexed all Emails, Web Cache documents, and all Files within user selected paths. The Web pages to be annotated have been selected randomly from each user's cache. To ensure that the user did not artificially collect a lot of data on the topic of the selected pages *after* having browsed them, we only pick pages browsed within the last week. For the annotation, we chose two input pages per each category: small (below 4 KB), medium (between 4 KB and 32 KB), and large (more than 32 KB) [FMNW03]. 96 pages were used as input over the entire experiment, and over 2,000 resulted annotations were graded.

Algorithms. We applied our three approaches for keyword extraction and annotation to the input pages, i.e., Document oriented, Keyword oriented and Hybrid. In all cases, in order to optimize the run-time computation speed, we chose to limit the number of output keywords extracted per Desktop document to the maximum number of annotations desired (i.e., four).

We applied the *Document oriented approach* with the following algorithms:

1. **TF, DF**: Term and Document Frequency;
2. **LC**: Lexical Compounds;
3. **SS**: Sentence Selection.

Second, we investigated the automatic annotation by applying the *Keyword oriented approach*. For step one of our generic approach (see Algorithm 2.6.2) we used the keyword extraction algorithms TF, DF, LC and SS. For step two we investigated the following algorithms for finding similar keywords:

1. **TC[CS], TC[MI], TC[LR]**: Term Co-occurrence Statistics using respectively Cosine Similarity, Mutual Information, and Likelihood Ratio as similarity coefficients;

[26] http://lucene.apache.org

2. **WN[SYN], WN[SUB], WN[SUP]**: WordNet based analysis with synonyms, sub-concepts, and super-concepts.

All in all, this gives us 24 possible combinations, i.e., TF+TC[CS], TF+TC[MI], ..., SS+WN[Sub], and SS+WN[Sup].

Finally, we conducted the study with the *Hybrid approach*. We used the four keyword extraction algorithms for both the input Web page and the identified Desktop documents. These were 16 combinations, i.e., TF+TF (applying TF for the Web page and TF for the Desktop documents), TF+DF, ..., SS+LC and SS+SS.

Rating the Personal Annotations. For each input page, the annotations produced by all algorithms were shuffled and blinded such that the user was not aware of either the algorithm which produced them, or of their ranking within the list of generated results. Each proposed keyword was rated 0 (not relevant) or 1 (relevant). It did not matter *how much* relevant each proposed annotation was, as it would have been a valid result anyway (i.e., if it has at least a reasonable degree of relevance to the input Web page, then it can be returned as annotation). It can be easily shown that those annotations closer to the actual content of the input Web document are generated more often.

We measured the quality of the produced annotations using Precision, a standard Information Retrieval evaluation measure. As our algorithms also generated a confidence score for each proposed annotation, we were able to order these suggestions and calculate the precision at different levels, namely P@1, P@2, P@3 and P@4. The precision at level K (P@K) is the precision score when only considering the Top-K output[27]. It represents the amount of relevant annotations proposed within the Top-K suggestions divided by K, i.e., the total number of annotations considered. Four different levels of K were analyzed, as it was not clear which is the best amount of annotations to generate using our approaches.

First, the P@K scores were computed for each user, algorithm, and Web page in particular. We averaged these values over the Web pages of the same type, obtaining user's opinion on the tuple <algorithm, type of Web page>. We further averaged over all subjects, and the resulting values are listed in the tables to come.

When doing the experiments, the users were specifically asked to rate the generated annotations taking into account both the target Web page and their personal interests. The idea was to generate for each subject not only generic keywords extracted from the current Web page, but others that connect it to his Desktop, and therefore his interests. Hence, the experiment is designed to evaluate our two main research questions: First, how well can automatic and personal annotation of Web pages be conducted with our approach? Second, which algorithm is producing the best results?

[27] In order to produce only highly qualitative results, we set some minimal thresholds for the confidence scores of each algorithm. Whenever they were not reached, the algorithm simply generated K' < K annotations for that input page.

Results

We will split our discussion of the experimental results in two parts, first analyzing the output of each approach in particular, and then comparing them to find the best method for generating personalized annotations. We compare our algorithms according to their P@3 scores (i.e., the average value over the ratings given to the Top-3 annotations, as ordered using their confidence levels), as most of them did not always produce more than 3 annotations above the strict minimal confidence thresholds we set.

Document Oriented Extraction. The investigation of our document based approaches provided us with an interesting insight: The TF metric performs best overall, thus generating better annotations than more informed techniques such as Lexical Compounds or Sentence Selection. After looking at each result in particular, we found that TF produces constantly good or very good output, whereas LC for example was much more unstable, yielding for some Web pages excellent annotations, but for some others rather poor ones. Nevertheless, both LC and SS had very good ratings, even the best ones for those subjects with dense Desktops. Thus, for future work one might consider designing an adaptive personalized annotation framework, which automatically selects the best algorithm as a function of different parameters, such as the amount of resources indexed within the personal collection. DF was only mediocre (except for the case of small input Web pages), which is explainable, as its extractions are based on generic entire Desktop statistics, thus diverging too much from the context of the annotated Web page.

Averaging over all types of input files, the best precisions were 0.79 when considering only the first output annotation, 0.81 when the top two results were analyzed, 0.79 for the top three, and 0.77 for all four. As LC yielded the first value, and TF the latter three ones, we conclude that LC is the method of choice if only one annotation is desired. However, its quality degrades fast, and TF should be used when looking for more results. All document oriented results are summarized in Tables 2.8, 2.9, and 2.10, for the small, medium, and large pages respectively.

Algorithm	P@1	P@2	P@3	P@4
TF	0.78	0.86	0.85	0.81
DF	0.83	0.76	0.76	0.74
LC	0.78	0.76	0.81	0.82
SS	0.67	0.75	0.75	0.73

Table 2.8 P@1-4 for document oriented extraction for small Web pages.

Keyword Oriented Extraction. We will first split the analysis of the keyword based algorithms based on its two steps. For extracting keywords from the single Web page used as input, all methods perform rather similarly for small pages, with TF performing slightly better. Again, we notice that the small input is the easiest one for our algorithms. More interesting, Document Frequency yields the best basis for keyword similarity search when used with medium and large input pages. It thus

Algorithm	P@1	P@2	P@3	P@4
TF	0.76	0.76	0.73	0.71
DF	0.59	0.51	0.55	0.53
LC	0.76	0.73	0.70	0.71
SS	0.76	0.67	0.68	0.67

Table 2.9 P@1-4 for document oriented extraction for medium Web pages.

Algorithm	P@1	P@2	P@3	P@4
TF	0.76	0.80	0.80	0.80
DF	0.63	0.63	0.60	0.54
LC	0.83	0.74	0.72	0.75
SS	0.82	0.74	0.71	0.69

Table 2.10 P@1-4 for document oriented extraction for large Web pages.

selects best those terms from the input Web page which are most representative to the user. This is in fact correct, as it is the only algorithm that relates the extraction also to the Desktop content by using DF values from the user's personal collection.

In the case of keyword similarity search, the results are very clear. WordNet based techniques seem to perform rather poorly, indicating external thesauri are not a good choice for annotations, as they cover too general a content. Note that one might obtain a better outcome when targeting this scenario to application specific thesauri and input Web pages. All co-occurrence based algorithms produce very good results, with TC[CS] and TC[MI] being slightly better.

The best average ratings overall were 0.81 at the first annotation (DF+TC[CS] and DF+TC[MI]), 0.82 at the second one (DF+TC[CS]) and finally 0.80, when considering only the top third output (DF+TC[MI]). No method managed to generally produce more than three highly qualitative annotations. All keyword oriented results are summarized in Tables 2.11, 2.12, and 2.13, for the small, medium, and large pages respectively. In order to ease the inspection of results, for each keyword extraction algorithm (i.e., first step), the best method is in bold.

Hybrid Extraction. These algorithms are composed of two phases, first a single page keyword extraction, and then another one, at the Desktop level and over those documents matching the best formerly extracted keywords (i.e., as obtained by searching the local Desktop with Lucene). All methods performed rather similar, with small differences between each other. An inspection of the actual data showed that again DF performed well at extraction in addition to extracting personalized keywords from the input page. However, this single word output was not always discriminative enough for a regular Desktop search, i.e., for finding Desktop documents highly similar to the input Web page. In fact, this is true for all keyword extraction techniques, and this is why the keyword oriented methods managed to surpass the hybrid ones in the quality of their output. For the second step, TF performed visibly better with medium and large pages, and all methods were close to each other for small input data. This is in accordance with the document oriented approaches, which use a similar technique.

Algorithm	P@1	P@2	P@3	P@4
TF+TC[CS]	**1.00**	**0.96**	**0.89**	**-**
TF+TC[MI]	0.83	0.83	0.89	-
TF+TC[LR]	0.75	0.88	0.89	-
TF+WN[Syn]	0.50	0.50	0.67	-
TF+WN[Sub]	0.41	0.41	0.37	0.18
TF+WN[Sup]	0.38	0.40	0.35	-
DF+TC[CS]	0.92	0.88	0.83	-
DF+TC[MI]	**0.92**	**0.88**	**0.86**	**-**
DF+TC[LR]	0.67	0.67	0.67	-
DF+WN[Syn]	-	-	-	-
DF+WN[Sub]	0.39	0.42	0.46	0.08
DF+WN[Sup]	0.67	0.53	0.48	0.20
LC+TC[CS]	**0.92**	**0.92**	**0.81**	**-**
LC+TC[MI]	0.92	0.88	0.81	-
LC+TC[LR]	0.65	0.70	0.74	0.65
LC+WN[Syn]	0.55	0.38	0.35	-
LC+WN[Sub]	0.26	0.34	0.36	0.24
LC+WN[Sup]	0.66	0.48	0.49	-
SS+TC[CS]	1.00	1.00	0.87	-
SS+TC[MI]	**0.83**	**0.83**	**0.89**	**-**
SS+TC[LR]	0.66	0.77	0.76	-
SS+WN[Syn]	0.43	0.43	0.43	-
SS+WN[Sub]	0.54	0.27	0.33	0.17
SS+WN[Sup]	0.53	0.47	0.37	-

Table 2.11 P@1-4 for kewyord oriented extraction for small Web pages.

SS+TF is the best overall algorithm, with a precision of 0.80 at only the first result, and of 0.74 at the top two ones. Then, TF+TF performs best, yielding a score of 0.73 at the top three annotations, and 0.74 when all results are included in the analysis. All results are also depicted in Tables 2.14, 2.15, and 2.16, for the small, medium, and large pages respectively. For each keyword extraction algorithm (i.e., first step), the best method is in bold.

Comparison. We now turn our attention to finding global conclusions over all our proposed algorithms. In order to better pursue this analysis, we depict the three best performing algorithms of each category (i.e., document oriented, keyword oriented, and hybrid) in Figure 2.10.

For small Web pages (the leftmost bar), the keyword oriented approaches are by far the best ones, being placed on positions one, two and four. They are followed by the document oriented ones, and finally by the hybrid methods, all global differences being very clear. In the case of medium sized pages, the proposed algorithms are harder to separate, performing similarly when analyzed over their best three representatives. Interesting here is the strong drop of TF+TC[LR], indicating that term frequency is good at single document keyword extraction only with small Web pages. Finally, we observe that the document oriented approaches are the best with large pages, which is reasonable, as they have the most amount of information available when searching the local Desktop for documents similar to the input Web page. They are followed by the keyword oriented techniques, and then by the hybrid ones.

Algorithm	P@1	P@2	P@3	P@4
TF+TC[CS]	0.64	0.64	0.64	-
TF+TC[MI]	**0.68**	**0.66**	**0.64**	**-**
TF+TC[LR]	0.70	0.66	0.63	-
TF+WN[Syn]	0.33	0.30	0.27	-
TF+WN[Sub]	0.41	0.39	0.36	0.08
TF+WN[Sup]	0.40	0.33	0.26	0.09
DF+TC[CS]	0.71	0.79	0.69	-
DF+TC[MI]	**0.71**	**0.75**	**0.75**	**-**
DF+TC[LR]	0.54	0.51	0.48	-
DF+WN[Syn]	0.15	-	-	-
DF+WN[Sub]	0.27	0.29	0.32	0.14
DF+WN[Sup]	0.38	0.30	0.27	0.10
LC+TC[CS]	0.60	0.63	0.57	-
LC+TC[MI]	**0.61**	**0.60**	**0.60**	**-**
LC+TC[LR]	0.59	0.61	0.58	0.35
LC+WN[Syn]	0.40	0.22	0.07	-
LC+WN[Sub]	0.31	0.41	0.36	0.09
LC+WN[Sup]	0.53	0.53	0.48	-
SS+TC[CS]	0.58	0.56	0.59	-
SS+TC[MI]	**0.62**	**0.60**	**0.60**	**-**
SS+TC[LR]	0.54	0.61	0.58	-
SS+WN[Syn]	0.23	0.13	0.09	-
SS+WN[Sub]	0.38	0.39	0.34	0.09
SS+WN[Sup]	0.33	0.30	0.18	-

Table 2.12 P@1-4 for kewyord oriented extraction for medium Web pages.

Figure 2.10 shows the quality of the output as a function of the input page size. As small Web pages contain few terms, they yield a clearer output, either when searching for related documents (as with document oriented techniques), or when extracting their keywords (as in the keyword oriented approaches), etc. As the content size increases, more noise appears, and processing becomes more difficult. Yet when Web pages have reached a reasonably large size, a number of informative terms tend to stand out, thus easing their processing to some extent.

We also averaged the results of all algorithms over all evaluated pages (in the rightmost column). We observed some obvious differences: (1) Keyword based algorithms (especially DF+TC[MI]), (2) Document based approaches, and (3) Hybrid ones. Though one would probably expect the latter ones to be the best, they suffered from the fact that the extracted keywords were insufficient to enable the retrieval of highly similar Desktop documents. On the contrary, the keyword based algorithms offer the optimal balance between the content of the input Web page and the personal files, producing a good selection of keywords which are both contained, as well as missing from the input page.

Finally, in Table 2.17 we give several examples for the output produced by the best two algorithms per generic approach. Let us inspect the first one in more detail. While some of the generated annotations can also be identified with previous work methods (e.g., “search”, as it has a high term frequency in the input URL), many others would have not been located by them either because they are not named entities and have

Algorithm	P@1	P@2	P@3	P@4
TF+TC[CS]	0.62	0.66	0.62	-
TF+TC[MI]	0.66	0.67	0.61	-
TF+TC[LR]	**0.65**	**0.68**	**0.64**	**-**
TF+WN[Syn]	0.25	0.08	0.05	-
TF+WN[Sub]	0.44	0.36	0.32	0.15
TF+WN[Sup]	0.44	0.46	0.33	-
DF+TC[CS]	**0.80**	**0.80**	**0.79**	**-**
DF+TC[MI]	0.80	0.77	0.79	-
DF+TC[LR]	0.85	0.71	0.70	-
DF+WN[Syn]	-	-	-	-
DF+WN[Sub]	0.44	0.38	0.37	0.02
DF+WN[Sup]	0.46	0.30	0.28	0.08
LC+TC[CS]	**0.57**	**0.61**	**0.58**	**-**
LC+TC[MI]	0.61	0.59	0.56	-
LC+TC[LR]	0.63	0.64	0.52	0.35
LC+WN[Syn]	0.31	0.19	0.16	-
LC+WN[Sub]	0.33	0.37	0.32	0.10
LC+WN[Sup]	0.53	0.43	0.39	0.06
SS+TC[CS]	**0.64**	**0.60**	**0.62**	**-**
SS+TC[MI]	0.60	0.60	0.60	-
SS+TC[LR]	0.64	0.63	0.61	-
SS+WN[Syn]	0.27	0.14	0.09	-
SS+WN[Sub]	0.51	0.40	0.38	0.14
SS+WN[Sup]	0.34	0.38	0.25	-

Table 2.13 P@1-4 for kewyord oriented extraction for large Web pages.

a low frequency in the input page (e.g., "schema" or "proximity search"), or simply because they are *not* contained in the starting URL (e.g., "retrieval" or "malleable" – both major research interests of our subject, highly related to the annotated page; or "Banks system" – a database search system very similar to the one presented in the given Web page; or "probabilistic", etc.).

Practical Issues. As we discussed earlier, the approach we propose is highly scalable: Our annotation algorithms are highly efficient, and utilize client processing power to annotate the Web pages. Users don't need to produce annotations for all pages they visit, but only for a sample of them (possibly randomly selected). The server collecting the data is free to limit the amount of incoming connections in order not to become overloaded.

From an implementation perspective, the algorithms proposed here can be easily integrated into browser toolbars already distributed by the major search engines. In fact, these toolbars already communicate with logging servers in order to send statistical browsing information[28], utilized for enhancing the search results ranking function. Thus, only a small additional communication cost is necessary.

[28]Only with user's consent.

Algorithm	P@1	P@2	P@3	P@4
TF+TF	**0.86**	**0.76**	**0.79**	**0.81**
TF+DF	0.92	0.71	0.69	0.69
TF+LC	0.75	0.79	0.78	0.73
TF+SS	0.92	0.79	0.75	0.76
DF+TF	0.83	0.88	0.83	0.77
DF+DF	0.92	0.71	0.69	0.67
DF+LC	0.67	0.67	0.64	0.63
DF+SS	**0.92**	**0.88**	**0.83**	**0.79**
LC+TF	0.86	0.74	0.74	0.76
LC+DF	0.92	0.71	0.69	0.63
LC+LC	**0.75**	**0.75**	**0.78**	**0.75**
LC+SS	0.86	0.72	0.73	0.74
SS+TF	0.89	0.78	0.77	0.79
SS+DF	1.00	0.75	0.72	0.65
SS+LC	**0.92**	**0.92**	**0.89**	**0.83**
SS+SS	1.00	0.79	0.78	0.72

Table 2.14 P@1-4 for hybrid extraction for small Web pages.

2.6.3 Applications

The approach we presented here enables a range of applications, from the most obvious, such as personalized Web search, Web recommendations to ontology learning and Web advertising.

Personalized Web Search. An interesting application is Web search personalization [CFN06a]. One could exploit such keyword extraction algorithms to generate term based profiles from each user's collection of personal documents. Upon searching some external collection (e.g., the Web), the output results could be biased towards those pages residing closer to the user profile in the terms hyperspace.

Web Recommendations for Desktop Tasks. It is quite common for people to search the Web for currently existing content to assist their present work tasks. It is possible to use the approaches we proposed in this paper in order to *automatically* suggest such pages [CFN06b]. More specifically, upon editing a Desktop document, relevant keywords would be extracted from the already written content and utilized to search on the Web for additional, useful resources on the same topic. Then, these automatically located pages could be displayed in a condensed format (e.g., title and URL) using a small discreet window placed for example in the bottom-right corner of the screen.

Ontology Learning. In the scenario of ontology-driven annotations, where an underlying ontology (customizable for each user) provides the terminology for such annotations, it might be necessary to enrich the ontology with user-specific concepts. These can be provided from the user's context, represented by keywords extracted from his Desktop environment. For instance, the initial personal information management ontology might lack a concept relevant to a specific user, for instance the class "Research Interests" as subclass of "Interests".

Algorithm	P@1	P@2	P@3	P@4
TF+TF	**0.71**	**0.76**	**0.74**	**0.74**
TF+DF	0.62	0.58	0.58	0.56
TF+LC	0.63	0.65	0.60	0.61
TF+SS	0.69	0.68	0.66	0.59
DF+TF	**0.70**	**0.66**	**0.59**	**0.59**
DF+DF	0.59	0.60	0.56	0.56
DF+LC	0.45	0.42	0.43	0.44
DF+SS	0.62	0.59	0.57	0.53
LC+TF	**0.73**	**0.78**	**0.74**	**0.71**
LC+DF	0.59	0.58	0.57	0.56
LC+LC	0.67	0.65	0.61	0.61
LC+SS	0.77	0.70	0.69	0.64
SS+TF	**0.80**	**0.76**	**0.70**	**0.68**
SS+DF	0.59	0.58	0.57	0.55
SS+LC	0.56	0.59	0.57	0.54
SS+SS	0.71	0.66	0.64	0.62

Table 2.15 P@1-4 for hybrid extraction for medium Web pages.

An analysis of keywords detected by the Lexical Compounds method is particularly valuable for an Ontology Learning approach, which we will investigate further. Based on the metrics described in this paper, the term "Research Interests" could be suggested as a keyword, and adopted as a class in the user-specific layer of the ontology. Relevant multiword expressions detected in such a way, sharing the same head noun, may be proposed as classes and organized hierarchically, while a hypothesis analysis for collocation over the Desktop corpus will enable us to distinguish between a simple modified head noun and a multiword expression bearing more meaning than the sum of its parts. Further knowledge may be extracted by considering sentences containing the keywords from the set determined by the Sentence Selection algorithm, which are often of descriptive nature. Considering the example above, it would be straightforward to identify "information extraction", "natural language processing" and "knowledge representation" as instances of the concept "Research Interests", given the beginning of the sentence "His research interests include information extraction[..]" and given the assumption that variations of this sentence will be found in documents on the Desktop and on the Web. This strategy will enable us also to extract instances, as well as relevant relations between the proposed classes.

Other Applications. If we move away from using personal Desktops, we can identify quite a lot of other applications of the same algorithms, some of them even already investigated. Due to space limitations we note here only one very important example: Web advertising. Keywords, as extracted from Web pages with the algorithms we presented, could be used to better match advertisements, as well as to propose better bidding expressions for the owner of the input Web site.

Algorithm	P@1	P@2	P@3	P@4
TF+TF	**0.67**	**0.68**	**0.68**	**0.67**
TF+DF	0.67	0.59	0.57	0.56
TF+LC	0.55	0.51	0.47	0.46
TF+SS	0.66	0.65	0.60	0.60
DF+TF	**0.52**	**0.62**	**0.66**	**0.62**
DF+DF	0.67	0.59	0.57	0.54
DF+LC	0.62	0.53	0.47	0.49
DF+SS	0.54	0.50	0.53	0.49
LC+TF	**0.58**	**0.63**	**0.63**	**0.63**
LC+DF	0.67	0.59	0.56	0.57
LC+LC	0.48	0.51	0.46	0.47
LC+SS	0.63	0.63	0.58	0.56
SS+TF	**0.72**	**0.69**	**0.63**	**0.64**
SS+DF	0.67	0.59	0.57	0.55
SS+LC	0.61	0.54	0.52	0.51
SS+SS	0.68	0.62	0.61	0.59

Table 2.16 P@1-4 for hybrid extraction for large Web pages.

2.6.4 Discussion

We have described a novel approach for scalable automatic personalized generation of annotation tags for Web pages. To the best of our knowledge there is no approach that does the same. Our approach overcomes the burden of manual tagging and it does not require any manual definition of interest profiles. It uses the implicit background knowledge on the users personal Desktop to propose personalized annotation tags for Web pages. In contrast to keyword extraction algorithms that can only propose terms that actually appear on the Web page, it proposes a more diverse range of tags which are closer to the personal viewpoint of the user. The results produced provide a high user satisfaction (usually above 70%). Thus, the greatest benefit of P-TAG is the high relevance of the tags for the user, and therefore the capacity of the tag to describe a Web page and to serve for a precise information retrieval.

We consider P-TAG as a valid step towards a lowercase semantic web, by the provision of personalized annotation tags for Web pages. We also see it as a valuable basis for the introduction of more semantics, i.e., for ontology learning approaches as mentioned in the Application Section (Section 2.6.3).

The current implementation of P-TAG is Desktop based. However for the near future we plan to implement a shared server approach that supports social tagging[29], i.e., the system would know about personal annotations from other users and would provide the most popular annotations, e.g., the ones with the highest score. This would enable the sharing of the automatic generated personal annotations in a collaborative environment, and would simply automatically create, apply and share tags dynamically.

For such a server based approach we envision the following advantages:

[29] http://en.wikipedia.org/wiki/Folksonomy

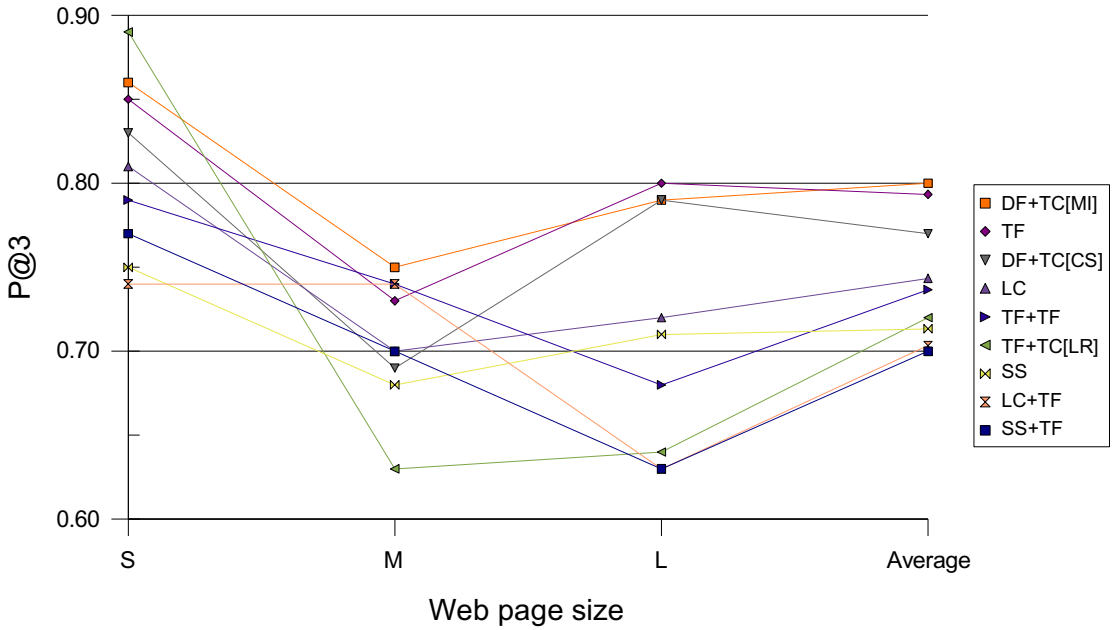

Figure 2.10 Precision at the first three output annotations for the best methods of each category.

1. *Diversity*: Keywords are generated from millions of sources, and thus cover various user interests.
2. *Scalability*: The annotation server can choose from which machines to collect the annotations, as well as from how many machines.
3. *High Utility for Web Advertising*: One can easily mine the dominating interests of the persons browsing a given Web page or set of pages.
4. *Instant Update*: We do not have to worry about the high volatility of the Web; newly created Web pages get annotated automatically, as they are visited by users.

We believe that P-TAG provides rather intriguing possibilities which can lead to a considerable high amount of annotated Web pages by automatic personalized tagging, thus lowering the barrier for the application of more semantics on the Web.

Algorithm	1^{st} **Annot.**	2^{nd} **Annot.**	3^{rd} **Annot.**	4^{th} **Annot.**
http://citeseer.ist.psu.edu/542246.html				
TF	Search	Information	User	System
LC	Proximity Search	Relational Databases	Banks System	Foreign Key
DF+TC[MI]	Schema	Search	Web	-
DF+TC[CS]	Schema	Search	Retrieval	-
TF+TF	Search	Query	Malleable	Schema
LC+TF	Database	Query	Probabilistic	Networks
http://en.wikipedia.org/wiki/PageRank				
TF	Web	Search	Information	Pages
LC	Search Engine	Web Pages	Authority Transfer	Link Structure
DF+TC[MI]	Web	Information	Conference	-
DF+TC[CS]	Web	System	Conference	-
TF+TF	Web	Page	Links	Semantic
LC+TF	Web	Search	Semantic	Information
http://www.l3s.de/ chirita/resume.htm				
TF	Bucharest	University	Search	Computer
LC	Computer Science	Information Retrieval	Personalized Web Search	Technical Report
DF+TC[MI]	Retrieval	Search	System	-
DF+TC[CS]	Retrieval	Search	Information	-
TF+TF	Search	Retrieval	Web	Ranking
LC+TF	Search	Web	Pages	Semantic

Table 2.17 Examples of annotations produced by different types of algorithms.

3 Ranking on the Desktop and on the Personal Virtual Information Space

3.1 Introduction

The PC Desktop, a clear reflection of the user and her activities, is now able to store hundreds of thousands of files that can be easily indexed. The complex human nature makes the structure of the stored data even more complicated and this materializes in the difficult task of finding one simple document on the hard-disk. Ironically, in quite a few of these cases nowadays, the file we are looking for can be found faster on the World Wide Web than on our personal computer. Existing desktop search systems, developed as a reaction to the rapidly increasing storage capacities of our hard disks, are an important step towards more efficient personal information management, yet they offer an incomplete solution.

Web search has become more efficient than PC search due to powerful link based ranking solutions like the PageRank algorithm [PBMW98] introduced by Google. 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. Indeed, desktop search engines are now comparable to the first generation of web search engines, which provided full-text indexing, but only relied on textual information retrieval algorithms to rank their results.

In this work we propose to analyze all user's activity patterns as an additional input for generating semantic links between desktop resources. This is a generalization over the methods we proposed in [CGNP06], in which only some special user actions are employed to establish semantical relationships on the desktop. Here, we first investigate and evaluate in detail the possibilities to translate this generic activity information into a desktop linkage structure, and then we propose several algorithms

that exploit these newly created semantic connections in order to efficiently rank desktop items. We also show that the ranking results based on the access links surpass both TFxIDF ranking scores, as well as the ranking schemes we proposed in our previous work, thus making them a valuable source of input to desktop search ranking algorithms.

In addition to the personal data, 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 - the Personal Virtual Information Space (VIS). Accessing these connected information sources in such a collaborative work environment becomes a crucial functionality, which so far has only been partially tackled within the new area of PIM. This has become a subject of growing interest [DH05, DHN+04], also to the database community, and (distributed and heterogeneous) dataspaces will extend databases beyond centralized and structured information repositories [FHM05]. 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 NEPOMUK[1] project [NEP] aimed 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 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 [CGNP05, CGNP06].

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 next 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.

Further, this work explores how we can use communication in social networks to

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

share and extend context information and how semantically rich recommendations between members of interest groups in such settings can be realized. We build upon FOAF networks, which describe personal and group information, based on the FOAF vocabulary to describe friends, groups and interests. We focus on how to share context in such a network, how to use these shared metadata to connect the information of different peers in the social network and how to use it for social recommendations.

After showing in the previous chapter how we tried to enhance the present structures on the desktop with various types of metadata - activity metadata considering time stamps, but also regarding the relationships between the resources used in various activities, in this chapter, we show how we can make use of all this metadata in order to enhance the search experience on the desktop. By trying to use the relationships between desktop resources, a desktop search system is already more useful to the user, but we try to move one step further, and offer a ranking mechanism other than classic TFxIDF on top of that. Thus, we go another step forward and apply additional ranking mechanisms, first using the additional time connections in Section 3.3, then explore how rankings are computed when the user exchanges resources within his community in Section 3.4, and also depending on the trust on his friends' resources, how ranking in influenced on the VIS in Section 3.5.

3.2 Related Work

Though *ranking* plays an important role on the Web, there is almost no approach specifically aiming at *ranking* desktop search results. Even if there exist quite a few systems organizing personal information sources and improving information access in these environments, few of them even discuss their search algorithms. In this section we briefly present some of these systems, then we continue with discussing several user activity studies for personal information, as they are useful for understanding people's behavior on the PC Desktops, and finally we describe some current ranking algorithms designed for use on the Semantic Web. Then we try to frame our work within the distributed environments' work, where we show some recommender systems, but also systems which attempt to do distributed ranking and also try to show how one community can influence the trust on a user's resources.

Desktop Organization Systems. Several search and retrieval tools make extensive use of the semantical relationships that can be inferred on the desktop, yet they do not employ any ranking scheme on top of this. Haystack [HKQ02, KBH+03] for example emphasizes the relationship between a particular individual and her corpus. It is quite similar to our approach in the sense that it automatically creates connections between documents with similar content and it exploits usage analysis to extend the desktop search results set. However, they do not investigate the possibilities to *rank* these results, once they have been obtained. Magnet [SK05] was designed as an additional component of Haystack with the goal to support naïve user

navigation through structured information via a domain-independent search framework and user interface. Gnowsis[2] adds Semantic Web interfaces to common desktop applications in order to link documents as within a personal semantic web. A number of adapters read data from different sources and make this information available as RDF. The created metadata is stored in a local RDF database and can be viewed, modified and searched (using basic TFxIDF ranking) with the aid of a Browser.

Stuff I've Seen [DCC+03] provides a unified index of the data that a person has seen on her computer, regardless of its type. Contextual cues such as time, author, or thumbnails can be used to search for and present information, but no desktop specific ranking scheme is investigated. Similarly, MyLifeBits [GBL+02] targets locally storing all digital media of each person, including documents, images, sounds and videos. They organize these data into collections and, like us, connect related resources with links. However, they do not investigate building desktop ranking algorithms that exploit these links, but rather use them to provide contextual information. Connections [SG05] is a very recent system also targeted at enhancing desktop search quality. Similar to us and to Haystack, they also connect related desktop items, yet they exploit these links using rather complex measures combining BFS and link analysis techniques, which results in rather large search response delays. Nevertheless, while our algorithms are clearly faster, we intend to compare the two approaches in terms of output quality in future work.

Finally, in prior work [CGG+05, CGNP06] we described our Beagle++[3] personal information system, a semantically enriched extension of the Beagle open source desktop search engine. There, we proposed various heuristics to generate ample activity based metadata associated to each desktop item. In addition, we generated links between resources in a similar manner to Haystack (e.g., between a file and an email, if the former resource was stored from the attachment of the latter), and we applied a Schema-Based PageRank [BHP04] to compute reputation scores. In this work we first formalize the desktop ranking process, and then we explore simpler, yet richer sources of linkage information between desktop items, such as global file access patterns.

Desktop Usage Analysis. Desktop usage behavior has been thoroughly analyzed in many studies. For example, Malone [Mal83] used interviews to analyze the way professional and clerical office workers organize information in their desks and offices. He identified two broad types of persons, *filers*, who organize their data into directories and categories, and *pilers*, who simply store all files in as few directories as possible. His work is orthogonal to ours, as we also analyze desktop user activity, but we focus on file access distribution, rather than storage behavior. Later, Barreau and Nardi [BN95] suggested that the way information is used on the PC Desktop should also be the primary determinant of the way it will be organized, stored and retrieved in the personal workspace. Finally, Jones et al. [JDB02] investigate the methods people use in their workplace to organize web information for re-use (e.g.,

[2] http://www.gnowsis.org/
[3] http://beagle.l3s.de/

send email to self, print out the web page, etc.).

Semantic Ranking. Aleman-Meza et al. [AMHAS03] analyzed the importance of semantically capturing users' interests in order to develop a ranking technique for the large number of possible semantic associations between 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. The approach is orthogonal to ours, as we build links only by exploiting fast usage analysis information, instead of using various complex algorithms to connect at run-time the desktop entities relevant for every specific user query. Another similar technique for ranking semantic query results is to analyze the inferencing processes that led to each result [SSS03]. In this approach, the relevance of the returned results is computed slightly faster than in the previous one, by exploiting the specificity of the relations residing within 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.

Recommender Systems. [DK04] presents a class of model-based recommendation algorithms for creating a top-N list of recommendations. In their approach, they first determine the similarities between the various items and then use them to identify the set of items to be recommended. [DK04] also addresses the key steps of this class of algorithms: which are the methods used to compute the similarity between items and which are the methods used to combine these similarities, in order to compute the similarity between a basket of items and a candidate recommender item. Opposed to this, in our approach the recommended items are based on user preferences and explicit context information.

Tapestry [GNOT92] is a recommender system which, in a sense, is similar to our approach. Tapestry is an e-mail filtering system, designed to filter e-mails received from mailing lists and newsgroup postings. Each user can write a comment / annotation about each email message and share these annotations with a group of users. A user can then filter these email messages by writing queries on these annotations. Though Tapestry allows individual users to benefit from annotations made by other users, the system requires an individual user to write complicated queries. We extend the idea in Tapestry by annotating not only emails but other resources on the user's desktop. In addition, exchange of annotations is handled (semi-) automatically.

The first system that generated automated recommendations was the GroupLens system [RIS+94]. The system, like in our case, provides users with personalized recommendations by identifying a neighbourhood of similar users and recommending the articles that this group of users finds interesting.

The most interesting work for recommendation infrastructures, which does not require a central recommender server is PocketLens [MKR04]. The paper discusses on how to preserve privacy in such an infrastructure. In contrast to our work, they do not exploit semantic connections between items, such as we have for citation relationships.

Compared to the usual recommender systems, including the commercial ones such as Amazon.com, which usually suggest single items, we have the potential to make semantically rich suggestions that are represented as parts of a semantic network which we exchange. Additionally we also provide to the user information about other users' rankings. While most recommender systems define groups by relying on the overlap among preferred items, we rely on an explicit group membership denotation based on FOAF metadata.

Distributed Ranking. In the last two years researchers have investigated how to compute PageRank in a distributed manner. [WD04] 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, [WA04] 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.

[CDKS01] 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, [PDMW06] 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 [BGS05].

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.

3.3 Ranking Using Activity Based Links

3.3.1 Context Based Ranking

A measure of importance for desktop resources is necessary in order to enable ordering these items within search. In this section we start with a description of the subset of the desktop ontology we use for transferring authority values, and then we formalize the structure of such a ranking mechanism based on the Google PageRank algorithm, as implemented in our Beagle^{++} system [CGNP06].

Desktop Ontology Overview

Given the fact that PageRank is inherently built on top of a linkage structure, we construct our algorithms on top of a subset of our desktop ontology which describes the relationships among the resources influencing the ranking. This is in fact similar to ObjectRank [BHP04], which introduced the notion of authority transfer schema graphs in order to propagate importance values across different nodes / contexts within all instances of the classes defined in some given ontology.

We explore three important semantical contexts in order to describe the impact of our activities upon our desktop: email, publications and web cache. We describe the semantics of these different contexts by appropriate ontologies (see [BCC$^+$06] for the complete ontology[4]). The links that describe these context specific connections are "departedTo", "cites", "hasSubject" / "isSubjectTo" and "contains" / "appearsIn". On the one hand, the semantic links generated this way are relatively few, as these particular events only occur a limited number of times. On the other hand, they do reflect a "clean" relationship in the sense that when a link is created, there most probably exists some semantic relationship between its components.

[4] http://www.kbs.uni-hannover.de/beagle++/ontology/

Exploiting Context Relations for Ranking

Following the approach described above, each user has her own contextual metadata graph, and for each node in this network the appropriate ranking can thus be calculated with the Schema-Based PageRank algorithm. This computation is based on the RDF link structure inferred over user's resources as specified in the desktop ontology and it takes the following form:

$$r = dAr + (1 - d)e \tag{3.1}$$

The vector r contains a score for each desktop resource, A is the normalized adjacency matrix which describes the graph connecting all available instances of the existing context ontology on one's desktop, and the e vector models a random jump to an arbitrary resource in order to guarantee convergence [PBMW98].

As an extension to our work from [CGNP06], we here also investigate several additional linkage heuristics, such as connecting two emails if they involve the same "bag" of persons (have the same sender and recipients), or if they have the same subject (threading of emails). Once all links have been generated, Schema-Based PageRank is applied as described above.

3.3.2 Activity Based Ranking

The contexts exploited up to now for ranking could also be interpreted as three different threads of activity: reading emails and their follow-ups, browsing through the file system for the cited papers from a publication, and browsing over previously visited web pages. However, they only address very specific user actions on the desktop. In this section, we extend this approach with a more general one, in which all accesses to desktop resources are considered for ranking.

Current personal information systems create links between desktop resources only when a very specific desktop usage activity is encountered (e.g., the attachment of an email is saved as a file, or a web page is stored locally, etc.). We argue that in fact in almost all cases when two items are touched in a sequence several times, there will also be a relation between them, irrespective of the underlying user activity (e.g., surfing the web, etc.), as well as if they were touched in a small window of time. Our ultimate goal is to infer links from desktop resources that have been accessed in a sequence or within the same time window. Yet this is not a straightforward task. Several persons might have quite different usage behavior patterns, thus making it very difficult to distinguish usage sessions from each other. More, this problem could even occur with the same person, at two different moments in time. We thus conducted a small usage behavior study in order to analyze users' file access patterns, more extensively presented in [GCC+08]. This allowed us to observe behavioral patterns and also led to several ideas we used here. Thus, we propose to add a link between two items a and b whenever item b is touched after a for the T^{th} time, with T being a threshold set by

the user. Higher values for T mean an increased accuracy of the ranking algorithm, at the cost of having a score associated to fewer resources. Theoretically, there is only a very low probability to have any two items a and b touched in a sequence even once. However, since *context switching* occurs quite often nowadays, we also investigated higher values for T, but experimental results showed them to perform worse than $T = 1$. This is in fact correct, since two files are accessed consequently more often because they are indeed related, than due to a switch of context.

To put this new heuristic into practice, we extend the ontology proposed in Section 3.3.1 with two additional links connecting two Desktop Documents: "Accessed in Sequence", if two items have been accessed in a direct sequence, and "Accessed in Window", which models the same heuristic in a more relaxed approach, namely two items being accessed within a small window of time (rather than in an exact sequence). When instantiated, these generated links amount to a much larger number than the context specific links we defined in previous work. They also come with the drawback of adding some additional noise, as their underlying heuristic is less strict against context switchings, but as we will see from the experimental results, it does indeed result in improved search quality, indicating that it adds more useful links than noisy ones.

Inferring Semantic Relationships by Analyzing Access Sequences

Another aspect that needs to be analyzed is the type of links residing on the PC desktop. In our approach we use directed links for each sequence $a \rightarrow b$, because if file b is relevant for file a, it does not necessarily mean that the reverse is true as well.

As it was not clear how many times two resources should be accessed in a sequence in order to infer a semantic connection between them, we studied several values for the T threshold, namely one, two and three. Additionally, we also explored the possibilities to directly use the original matrix A with PageRank, thus implicitly giving more weight to links that occurred more frequently (recall that in A each link is repeated as many times as it occurred during regular desktop activity).

Inferring Semantic Relationships by Analyzing Activity Sessions

The second method we tested is based upon the observations made by Soules et al. [SG05], stating that files accessed in a small time window (e.g., 30 seconds) are usually related. Therefore, we connect them in a similar manner like above: If the difference in access time stamps between files a and b is below a threshold, then we connect them, no matter whether they are in a sequence or not. This is in fact another approach to deal with context switches: Within a window, even though noisy context switching links are added, we also add the correct ones, describing real semantic relationships, the assumption being that there will be a lot more correct links than noisy ones. Finally, we also take into account the number of occurrences of each generated link,

especially in order to avoid the appearance of such noisy links.

3.3.3 Experiments

Experimental Setup

We evaluated how the newly introduced activity-based ranking algorithms perform when compared to TFxIDF, as well as to our previous work (formalized in Section 3.3.1). The experiments were performed within the research environment of L3S, on a total number of 10 persons, who installed an activity logging tool and worked normally on their desktops for several months. At experimenting time, they performed several desktop searches related to their regular activities, and graded each top-5 result of each algorithm with a score ranging from 0 to 2, 0 defining an irrelevant result, 1 defining a relevant one, and 2 a very relevant one. The grades were then averaged for each query and each ranking algorithm, and then again per subject for each algorithm over all queries. This gave us one grade per algorithm per subject.

The two global conventions we took for all ranking algorithms were as follows: First, an activity session was over after one hour of break. This was necessary in order to avoid linking documents accessed within different days, etc. In future work, we also intend to investigate more complex definitions of usage sessions, which for example relate to the average break time of a user, rather than to a fixed amount of time. Second, we investigated how our algorithms perform using a real desktop search scenario, i.e., with ranking scores combined with term frequency information. We used the following formula:

$$Score(file) = NormalizedLinkageScore(file) * NormalizedVSMScore(file, query)$$

The Linkage score is computed by each of our algorithms, and the VSM score is computed using the Vector Space Model and is based on TFxIDF. Both scores are normalized to fall within [0,1] for a given query[5].

Each subject prepared 6 queries: One set of single-word queries, and a second set of multiple-word queries. Each set contained three queries: a very precise one, a semi-ambiguous one, and finally an ambiguous one. For each of them the results were ranked with the seven algorithms summarized next, plus the standard TFxIDF ranking method. For every query, we shuffled the top-5 URIs output results for each algorithm, such that the users were neither aware of their actual place in the rank list, nor of the algorithm(s) that produced them. On average, for every issued query, a subject had to evaluate about 20 documents, for an average dimension of the desktop data of about 7.500 documents. In total, 60 queries had been issued and around 1,000 documents were evaluated.

[5]In order to avoid obtaining many null scores when using access frequency or total access time (recall that many items have never been touched by the user), in these scenarios we also added a $1/N$ score to all items before normalizing, with N being the total amount of desktop items.

Experimental Results & Comments

The base reference for comparing our ranking methods is TFxIDF, as it is the most widely used approach in desktop search. Our rankings are all obtained by applying the Schema-Based PageRank described in Section 3.3.1 on a matrix representing various linkage heuristics between resources. This ranking procedure gives us a document reputation score, which is stored as an additional RDF property for each resource. Then, at run-time, this score is multiplied with the TFxIDF one, thus obtaining the final score for each search result. All our tested algorithms are summarized below:

1. **C - Context**: As defined in Section 3.3.1, the link matrix representing the contextual links defined in our previous work.
2. **SA - Sequential Activity**: As described in Section 3.3.2 (Access Sequences), with an occurrence threshold $T = 1$.
3. **WA1 - Window Activity 1**: As described in Section 3.3.2 (Activity Sessions), with a window size of 30 seconds and an occurrence threshold $T = 1$.
4. **WA2 - Window Activity 2**: Same as above, but with an occurrence threshold $T = 2$.
5. **CSA - Context & Sequential Activity**: A combination of the links generated by the two approaches.
6. **CWA1 - Context & Window Activity 1**: Same as above, but for Context and Window Activity 1.
7. **CWA2 - Context & Window Activity 2**: Same as above, but for Context and Window Activity 2.

We present in Table 3.1 the averaged grades over all queries for each of the 10 subjects, as well as the average over all subjects and the standard deviation for each ranking algorithm. As our group of testers was relatively small, we also employed two methods to eliminate outliers. The first one is a threshold of 0.75 on the relative error to the average, which designated subjects 3, 4 and 6 as outliers. The second is a threshold of 3.5σ (with σ the empirical standard deviation) on the z-score of each value, which removed subjects 1, 2, 5, 7, 8, 9 and 10. Note that if a subject's grade of one algorithm is detected as outlying, then all the grades for this subject are ignored (to ensure uniformity of the results). As the z-scores method removed too many observations, we will proceed our analysis only for the results obtained using the relative error to the average as an outlier detection method.

The graphs of the average scores obtained without outliers are plotted in Figure 3.1. They show that all semantic ranking algorithms (context, activity, and combinations) perform much better than the simple TFxIDF scoring, some of them even close

Subject ID	TFx IDF	C	SA	WA1	WA2	CSA	CWA1	CWA2
1	0.667	0.900	1.000	0.767	0.967	0.933	0.867	0.900
2	0.900	0.867	0.900	0.867	0.833	0.867	0.900	0.800
3	0.433	0.500	0.433	0.500	0.333	0.500	0.542	0.500
4	0.567	0.567	0.633	0.467	0.600	0.500	0.533	0.567
5	0.833	0.800	0.800	0.900	0.667	0.700	0.800	0.800
6	0.500	0.600	0.800	0.633	0.700	0.667	0.600	0.600
7	0.200	0.500	0.200	0.133	0.200	0.267	0.233	0.367
8	0.233	0.367	0.200	0.167	0.300	0.200	0.167	0.267
9	0.167	0.233	0.167	0.167	0.267	0.400	0.367	0.233
10	0.633	0.533	0.767	0.767	0.900	0.700	0.600	0.533
STD	0.0178	0.0427	0.0255	0.0260	0.0213	0.0263	0.0243	0.0101
Avg Err	**0.504**	**0.563**	**0.600**	**0.546**	**0.592**	**0.575**	**0.559**	**0.550**
T-Test Err	N/A	0.077	0.054	0.113	0.090	0.104	0.100	0.116
Avg Z	0.500	0.556	0.622	0.533	0.544	0.556	0.558	0.556
T-Test Z	N/A	0.100	0.156	0.339	0.330	0.249	0.166	0.100

Table 3.1 Algorithms' normalized grades averaged over all queries. STD: Standard deviation on all observations. Avg Err: Average over the inliers according to relative error threshold. T-Test Err: T-Test relative to the TFxIDF grades for the inliers according to relative error threshold. Avg Z: Average over the inliers according to z-scores threshold. T-Test Z: T-Test relative to the TFxIDF grades for the inliers according to z-scores threshold.

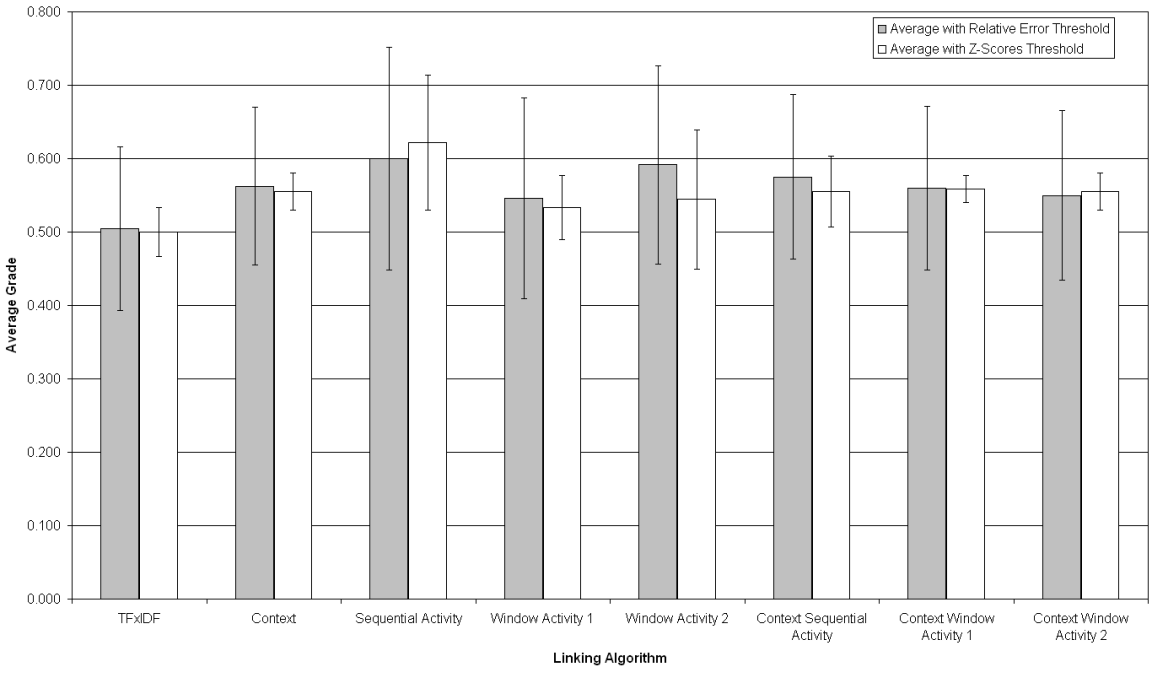

Figure 3.1 Average grades per algorithm with standard deviation.

to statistically significant levels (e.g., t-test score of 0.054 for Sequential Activity). All in all, the Sequential Activity algorithm performs best, the second one being Window Activity 2, very close to our previous work, Context. It was however surprising to find out that the combinations of these algorithms do not improve the results. We think this may have several reasons: First, the activity metrics, as they generate a lot of links, may have already covered the "contextual" links, and thus no additional useful information would be added by combining them; Second, the amount of links generated by the activity based heuristic may be a lot bigger than the amount of links generated by the contextual approach, thus neutralizing these additional links. Further, more thorough investigations are necessary in order to identify the right cause. However, we can safely conclude that the activity based heuristics do indeed yield better results when used alone for ranking desktop items.

3.3.4 Discussion

In the previous sections we proposed to use activity based heuristics in order to rank desktop search results. We first formalized our previous Schema-Based PageRank approach to generate resource rankings based on several specific user activities. Then, we proposed an additional extension to the underlying schema, according to which two items are linked if they have either been accessed in a sequence, or within a window of time. We investigated in detail several approaches to create this kind of links, and compared them to our previously defined heuristics, as well as to TFxIDF ranking. Our empirical results showed that usage analysis is indeed an improvement over both the context based and the TFxIDF ranking, thus making it a valuable source of input to desktop search ranking algorithms.

In future work we intend to explore content based heuristics to provide us with further additional links between similar desktop documents. Also, we would like to analyze the necessity and benefits of enabling desktop search restrictions to only some specific sub-tree of the local file hierarchy. Finally, we intend to devise several methods to detect activity contexts based on activity analysis, and then integrate them into the ranking scheme itself.

3.4 Sharing, Exchanging and Ranking Semantic Context Based on Recommendations

In the previous section, we showed how links between resources on the desktop can be exploited for providing the user a ranking scheme. We now focus on a distributed scenario, where a user shares his resources with the people in his social groups, and his friends at their turn share theirs. The user also receives recommendations about useful resources and provides his own to the others. In such scenario, computing a rank for the received resources becomes complicated, especially because of the trust

the user has on the persons that recommend him resources, and which need to be taken into account. We next provide such a solution for a ranking scheme in a distributed scenario.

3.4.1 Motivating Scenario

As our motivating scenario, let us consider our L3S Research Group context and within this group, Bob and Alice as two members who exchange information. One important task in a research group is exchanging and sharing knowledge, which we will focus upon in this section. Unfortunately, the most widely used infrastructure for this purpose, email, is poorly suited to support this exchange. When we exchange documents by email, no context is shared (for example which are the interesting follow-up papers, or which are the interesting references for a paper) and any comments about the documents that are included in the email are lost as soon as the attached documents are stored in some directory.

The following example shows how such a sharing scenario can be supported in a more efficient manner. We assume that Bob mails Alice a document which he sent to the DELOS Workshop, with the title "I know I stored it somewhere - Contextual Information and Ranking on Our Desktop". Bob is one of the authors and therefore he already has all the important context for this paper including the cited papers stored on his computer. In this first email, Alice will therefore not only receive the paper but also its immediate context relevant for the research group, containing information about all papers that are referenced in the DELOS paper, information about important authors for this topic or which conferences are relevant. In other words, whenever we send a paper, the metadata associated to that paper will also be sent. From the five references included, Alice decides that "ObjectRank: Authority-Based Keyword Search in Databases" and "Activity Based Metadata for Semantic Desktop Search" are of particular interest for her and she sends back an email to Bob requiring additional information about those. As an answer, she receives from Bob the context information associated with these papers, containing the references that Bob has already downloaded. So the context information will be exchanged progressively, from the immediate context to the more distant one.

Figures 3.2 and 3.3 present the context created on Alice's desktop as result of her metadata exchange with Bob. Figure 3.2 contains only the *cites* relationship among the various resources, while in figure 3.3 we represent additional relationships, like *presented_at*, *downloaded_from*, *author*, or *same_session*. Note that the context networks created on the users' desktop are not separated, but just visualized separately in these figures.

By examining the context graph in figure 3.3, we see that all the papers labelled from **G** to **Q** were presented at different WWW Conferences, in different years, and all were downloaded from the ACM Portal. Papers **A**, **C** and **K** all share the same author, *Bob*, and have been downloaded from the L3S Publication page. Similarly,

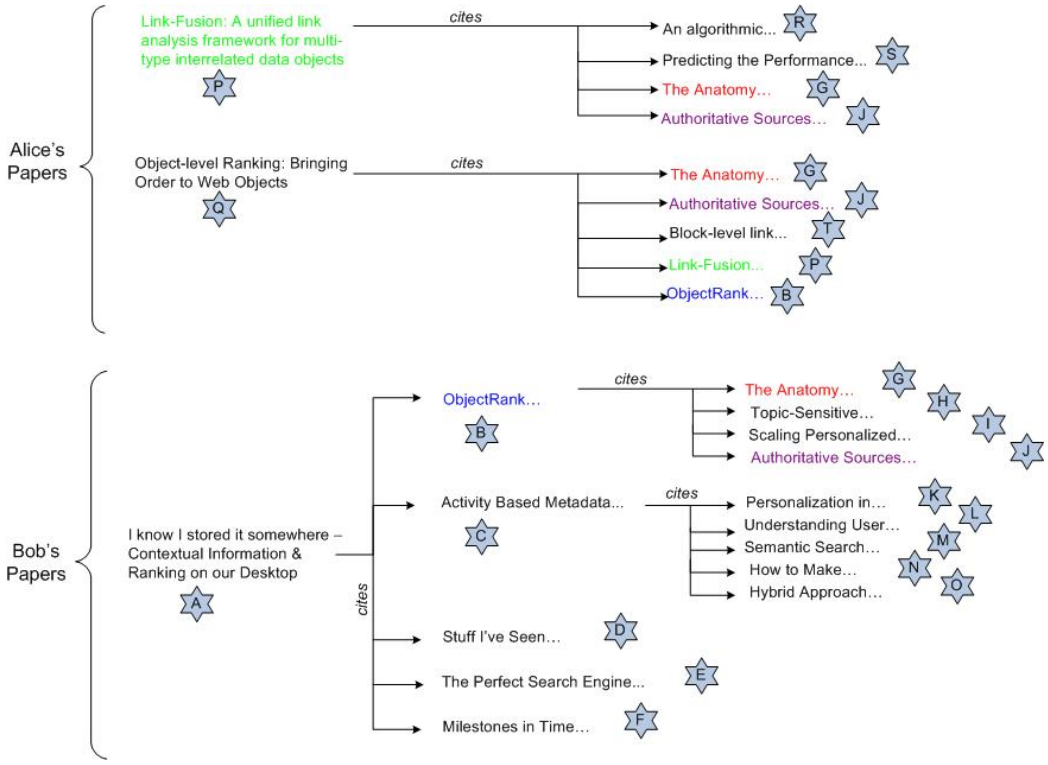

Figure 3.2 Publications Context Example - Part 1

the publication labelled **B** and the other two papers which were presented at the same session at the VLDB conference were downloaded from the VLDB web site.

All this information is taken into account when computing the importance of the resources on Alice's desktop. For example, when computing the importance of the conferences, the WWW Conference will be more important than other conferences, since Alice already has a lot of important publications which have been presented there. The number of papers from the same author Alice has already downloaded also influences how important she considers that author. This means that certain authors are more important than others, based on the publications used and cited in the L3S Research Group, as well as on general citation information about these authors. The fact that Alice knows Bob and Bob is one of the authors of three publications Alice has on her desktop influences the importance of Bob's publications and of course, Bob's importance as author. So he will be definitely more important to Alice than other authors not known to her.

In order to be able to compute the rankings of their documents Alice and Bob have to build a context around the resources they have stored on their desktops. The next section presents in more detail how this context information is created and then describes how this context can be used in computing rankings of search results on the desktop [GNP05a].

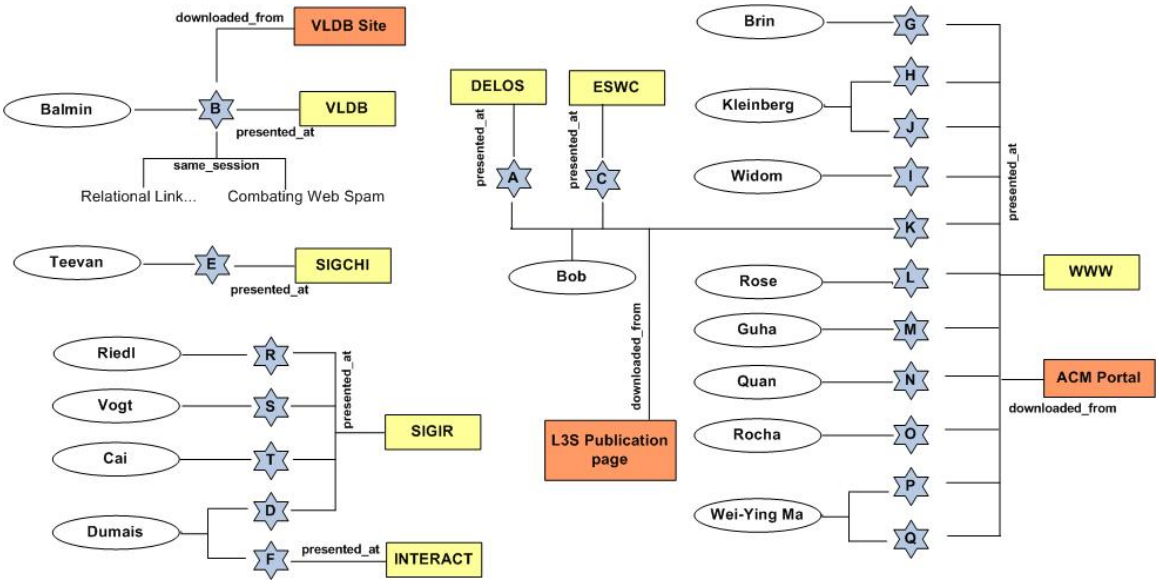

Figure 3.3 Publications Context Example - Part 2

3.4.2 Representing Context and Importance

Representing Context

Generally speaking, context information describes all aspects important for a certain situation: ideas, facts, persons, publications, and many more. Context information includes all relevant relationships as well as interaction history. Current desktop search prototypes fall short of utilizing any desktop specific information, especially context information, and just use full text index search. In our scenario we clearly need to use additional context information, and specifically want to exploit the following contexts:

CiteSeer context. The most important aspects we want to record from the CiteSeer context are the publications we are viewing or downloading and how these publications are connected to other publications. Important parts of the available context information are the authors of these publications, the conferences in which they were presented or the year when they were published and even more, the publications which cite them or are cited by them. We want to keep track whether we saved a certain publication on our own desktop in order to be able to find it later and we want to receive suggestions about papers that might be interesting in the same or overlapping contexts.

CiteSeer provides four additional types of links that can be followed after identifying a paper. The most expressive in our case would be the ones that refer to the related documents from co-citations and the papers that appear on the same web site.

Browsing and Desktop context. Browser caches include all information about user's browsing behaviour, which are useful both for finding relevant results, and for providing additional context for results. In our scenario, when we search for a document we downloaded from the CiteSeer repository, we do not only want to retrieve the specific document, but also all the referenced and referring papers which we downloaded on that occasion as well.

In general, we view documents stored from emails and from web sites as our personal digital library, which holds the papers we are interested in, plus all relevant contextual information. When we store documents, we can then retrieve them efficiently and restore the original context we built up when storing these documents. Personalized search and ranking on the desktop takes this contextual information into account as well as the preferences implicit in this information. [TAAK04] discusses how people tend to associate things to certain contexts. So far, however, search infrastructures neither collect nor use this contextual information.

Scenario specific annotation ontologies. Figure 3.4 presents our current prototype ontology, used for implementing our motivating scenario. It specifies context metadata for the CiteSeer context, files and web pages, together with the relations among them (described in more detail in [CGG+05]). Conceptually, the elements in the rectangles represent classes, circles represent class attributes. We use classes whenever we want to attach importance / rank on entities, attributes otherwise.

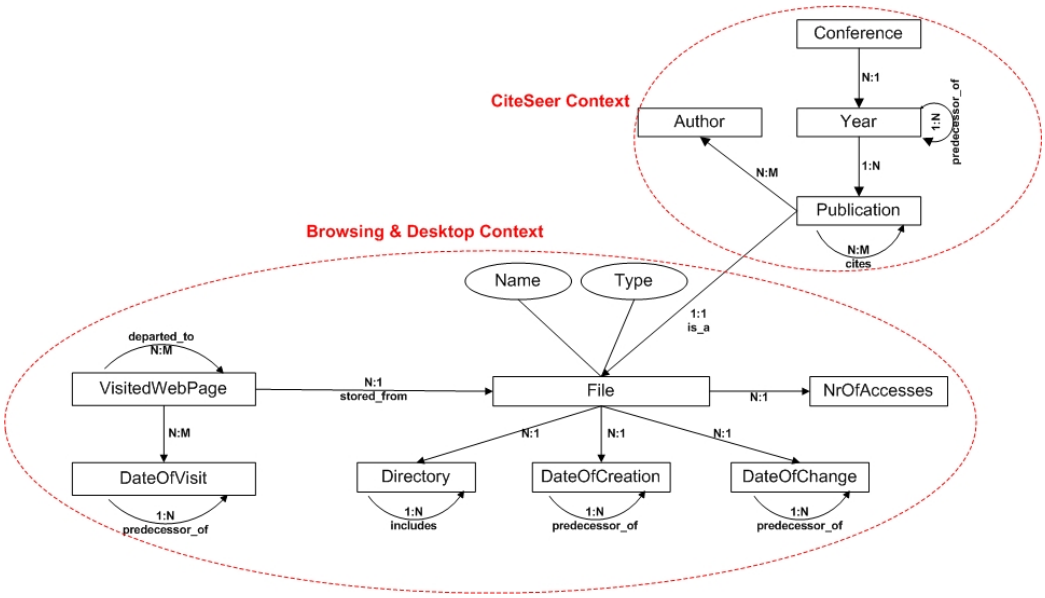

Figure 3.4 Context ontology for our prototype

For the browsing and desktop context, we annotate each page with additional information about its basic properties (URL, access date, etc), as well as more complex ones such as in- and out-going links browsed ([CGG+05]). The user's behaviour as the

pages or publications he browsed or downloaded provide useful additional information. Files, which are stored from web pages, reside in certain directories, which in turn can include other directories. The creation or change date of a file together with the number of accesses are some other important indicators which have to be taken into account when describing the desktop context. An extended publication ontology makes use of additional knowledge about how CiteSeer pages are connected and what they represent. Publications are referenced by other publications and can cite others, they can have a publication date / year associated with them, as well as a conference or journal. Publications have authors and are stored as documents on the desktop.

Other ontologies describe contexts like conferences, including reviewers, papers, meetings, authors, or private contexts like birthdays, including persons, locations, etc.

Representing Importance

In addition to the information which resources are included in a specific context, we also want to know how important or valuable these resources are. We therefore have to develop a mechanism which allows us to express this information and use it for ranking search results.

Authority transfer annotations. Annotation ontologies describe all aspects and relationships among resources which influence the ranking. The identity of the authors, for example, influences our opinion of documents so "author" should be represented explicitly as a class in our publication ontology. We then have to specify how these aspects influence each other's importance.

ObjectRank [BHP04] 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 the 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.[6]

Figure 3.5 depicts our context ontology plus its authority transfer annotations. The ontology representing our browsing and desktop context says that a visited web page is important if we arrived at the current one from an important page, if the file under which it is stored is important, or if the date when the page was visited is important. For the CiteSeer context, publications transfer part of their authority to other papers they cite, to their authors, to the files under which they are stored, and to the year when the paper was published. As we can see, citing important papers doesn't make a paper important. As suggested in [BHP04], every edge from

[6] In contrast to ObjectRank, we do not compute a keyword-specific ranking, but a global one.

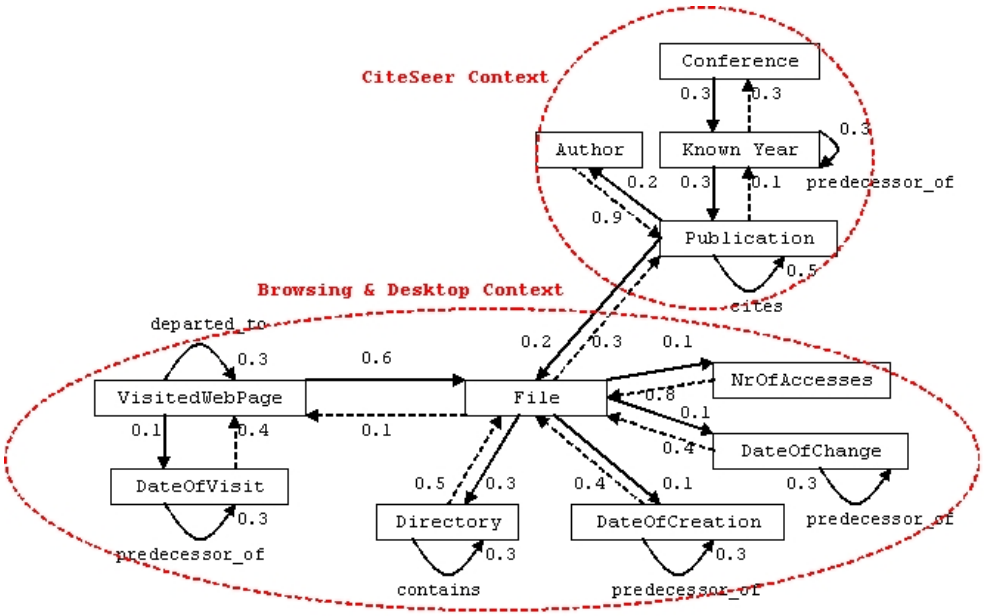

Figure 3.5 Authority transfer annotations, including external ranking sources

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.

Personalized Preferences and Ranking. 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 user ontology all concepts, which influence our ranking function. For example, if we view a publication important because it was written by an author important to us, we have to represent that in our context ontology.

3.4.3 Sharing Context and Importance

Interest Groups

Interest groups in our context are specialized social networks that have a stated common interest which connects the members of the group. One important reason for creating interest groups resides in increasing the efficiency of the information flow inside that group. All members of the same interest group share the same domain of

interest and the social relationships are woven around this type of information sharing. They are all possibly part of the same professional group, just as we described in the motivating scenario, Alice and Bob being in the same research group, the L3S Research Group.

We chose to represent interest groups based on an extension of FOAF in order to describe the social network of participants and we will describe all contexts as RDF metadata, as presented in [CGG+05]. Being based on RDF, FOAF inherits some of its benefits, like the ease of aggregating and harvesting it, or combining it with other vocabularies, thus allowing us to capture a rich set of metadata. The basic FOAF vocabulary itself is pretty simple, pragmatic and designed to allow simultaneous deployment and extension. It is identified by the namespace URI 'http://xmlns.com/foaf/0.1/' and described in more detail at the FOAF project page [The].

FOAF terms represent information which can be grouped in the following five broad categories: FOAF Basics, Personal Information, Online Accounts/ IM, Projects and Groups, Documents and Images. The most important for us is the *Projects and Groups* category, which allows us to talk about groups and group membership among others. Groups are represented with the aid of the **foaf:Group** class, which represents a collection of individual agents. The **foaf:member** property allows us to explicitly express the membership of agents to a group. Since the **foaf:Person** class is a sub-class of the **foaf:Agent** class, persons can also be members of a group. One can specify the interests of the group members by using specific properties, like **foaf:interest**, **foaf:topic_interest**, or **foaf:topic**, even though it is not yet clear how to use them correctly.

A notable omission in the basic FOAF vocabulary is the inability to express anything related to information sharing in a group. Even though being in a group or social network usually means that we want to share information within this social network, there is no vocabulary to express this in FOAF. The assumption we make in this paper is that people belonging to a common interest group will share a specific set of metadata. In our scenario these are the contextual metadata defined by appropriate annotation ontologies, as discussed in the previous section. When members of an interest group express that they want to share a certain set of metadata, they will agree on an appropriate ontology defining this set. We will therefore extend the FOAF vocabulary with a new property **foaf:shared_context** which takes as its value the annotation ontology describing the metadata to be shared. Based on this, the FOAF description of the L3S Research interest group and its members Bob and Alice as presented in our motivating scenario looks as follows:

```
<foaf:Group>
  <foaf:name>L3SResearchGroup</foaf:name>
  <foaf:member>
    <foaf:Person>
      <foaf:name>Alice</foaf:name>
      <foaf:homepage rdf:resource="http://www.l3s.de/~alice"/>
    </foaf:Person>
  </foaf:member>
  <foaf:member>
    <foaf:Person>
      <foaf:name>Bob</foaf:name>
      <foaf:homepage rdf:resource="http://www.l3s.de/~bob"/>
    </foaf:Person>
  </foaf:member>
  <foaf:shared_context
  rdf:resource=http://www.l3s.de/isearch/citeseerContext.rdf/>
  <foaf:shared_context
  rdf:resource=http://www.l3s.de/isearch/browsingDesktopContext.rdf/>
</foaf:Group>
```

Exchanging Context within Interest Groups

Sharing context in an interest group is useful and necessary because not only do we want to publish our own work but we also want to find out about additional new resources related to our work and get suggestions about possible further developments in that area. Recommendation then means suggesting additional related information to given items. In our motivating scenario, we have as interest group a set of researchers, and a set of ontologies defining which metadata are shared between them. The contextual metadata corresponding to those ontologies as discussed in section 3.4.2 represent the context information we have available on our desktop.

These context metadata are generated locally by a set of metadata generators [CGG+05], which record user actions as well as interactions and information exchanges between members of a group. These metadata generators create RDF annotation files for each resource whose context they describe, so for each relevant resource on the desktop (e.g. a specific publication) we will have this additional RDF information available.

For the experiments described in this paper, we have implemented a metadata generator, which deals with publications, and crawls one's desktop in order to identify and annotate all papers saved as PDF files. 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 annotation file, containing information from the database about the title of the paper, the authors, publication year, conference

and other CiteSeer references to publications. All annotation files corresponding to papers are then merged in order to construct the RDF graph of publications existing on one's desktop.

In our scenario, whenever Bob sends a publication to Alice, who is member of the same interest group, he wants to attach the appropriate context information, i.e. the publication context we have discussed in the scenario. A second (email) helper application therefore checks who is the recipient of the email, which group she belongs to, and therefore which context information/ metadata to attach. On Alice's side, the helper application has to integrate the newly received annotation files into the existing publication graph.

Sharing Importance

Ranking of Resources - General Algorithm

In our distributed scenario, 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 3.4.2. The computation of rankings on one's desktop is based on the link structure of the resources as specified by the defined ontologies and the corresponding metadata. When sharing information within the group / network we exchange not only contexts but also rankings. So exchanging context information has also an impact on the ranking of results of the desktop search. These values are then recomputed according to the rankings received together with the context from other persons.

Ranking of resources is calculated based on the PageRank formula:

$$r = dAr + (1 - d)e \tag{3.2}$$

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 modelled by the vector e. 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. Thus, 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.

To make these details clear, let us look at the following example: we consider the authority transfer annotation ontology for a publication ontology, as depicted in Figure 3.6 and then instantiate it. A subset of this data graph is shown in Figure 3.7.

The instantiation of our the matrix A from Equation 3.2 is depicted in Table 3.2:

According to Figure 3.7, the authors transfer 0.5 units of importance to their own publications (Wei-Ying Ma to publications P, Q, Kleinberg to H and J and Balmin to

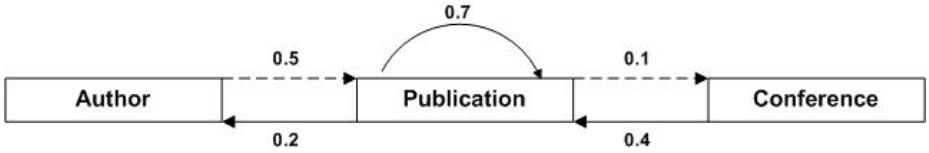

Figure 3.6 Authority transfer annotation ontology for a publication ontology

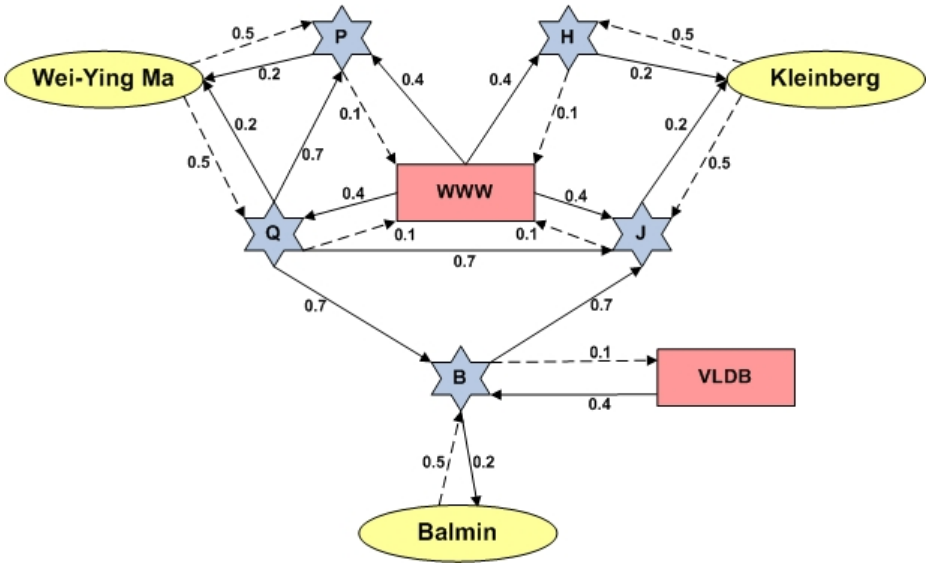

Figure 3.7 Data Graph

B). Publications transfer 0.7 units of importance to other papers they cite, 0.1 units to the conferences where they were accepted and 0.2 units of importance to their authors.

The values in the matrix are grouped in blocks, formed by considering the cartesian products *Author* × *Author, Author* × *Publication*, etc. The elements from one block 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 between the entities which determine the block, from the authority transfer annotation ontology.

Ranking of Resources on Alice's Desktop

We computed the ranking values for the resources existing on Alice's desktop (see Figure 3.2 for the labels of resources). The results are presented in Table 3.3. Note that these values represent Alice's personal rankings according to the context existing around her resources and are not necessarily related to external sources of ranking like CiteSeer or Google.

How Ranks Change When Bob Sends Something

After receiving via email the context existing on Bob's desktop, as we described in Section 3.4.1, Alice's ranks change as presented in Table 3.4. By comparing the values in the two tables, we can see that the rankings increase, because existing resources

$$A = \begin{array}{c} \\ YingMa \\ Kleinberg \\ Balmin \\ P \\ Q \\ J \\ H \\ B \\ WWW \\ VLDB \end{array} \begin{array}{c} \begin{array}{cccccccccc} Y.Ma & & & & & & & & & VLDB \end{array} \\ \begin{pmatrix} - & - & - & 0.2 & 0.2 & - & - & - & - & - \\ - & - & - & - & - & 0.2 & 0.2 & - & - & - \\ - & - & - & - & - & - & - & 0.2 & - & - \\ 0.5 & - & - & - & 0.7 & - & - & - & 0.4 & - \\ 0.5 & - & - & - & - & - & - & - & 0.4 & - \\ - & 0.5 & - & - & 0.7 & - & - & 0.7 & 0.4 & - \\ - & 0.5 & - & - & - & - & - & - & 0.4 & - \\ - & - & 0.5 & - & - & - & - & - & - & 0.4 \\ - & - & - & 0.1 & 0.1 & 0.1 & 0.1 & - & - & - \\ - & - & - & - & - & - & - & 0.1 & - & - \end{pmatrix} \end{array}$$

Table 3.2 The A matrix

are referenced by the newer ones. For example, the rank of the "ObjectRank" paper, labelled **B**, increases from 0.594175 to 1.124971 since it is now referenced by the paper labelled **A**. As a consequence, all the rankings for the resources which have an incoming link from **B** will increase. This process of rank propagation is an iterative one, according to the links in the data graph, and continues until the rank difference between two iterations is less than a certain threshold. Alice receives not only context from Bob, but also resources, so that she will also have rank values for these new resources.

The context which is received from other members of the interest group is used for building the user's own context, which means that it is also taken into account when creating the adjacency matrix A. In order to include the rankings of other users into the computation of the user's own ranking, we work on the vector e, which models the random jump. So, if a resource is highly ranked according to the received rankings and the user wants to take this into account, she will have to assign a higher value for the corresponding element in the vector which simulates the random jump.

Of course, even if two users exchange all of their context metadata, they still will not have the same rankings, as local usage information such as number of accesses etc., which influences rankings, always stays local and is not exchanged. Note that in our data graph, group members usually appear as instances of authors or as senders of emails [CGG+05], so we can use their rank as one possible indicator of their trustworthiness.

How Alice's Trust in Bob Influences the Rankings

Even inside an interest group, we have to take into account different reputations. If somebody, whom I trust and who is important for me, sends his recommendations, I want his suggestions to be higher ranked than the ones received from a more untrusted person. These different reputations can be represented by influencing the dampening factor. The higher the trustworthiness of someone in my interest group, who sends me her own context and rankings, the higher should be the probability to reach the resources in that set.

Resource	Rank
W. Y. Ma	0.349624
Riedl	0.289991
Vogt	0.289991
Cai	0.260351
Balmin	0.251222
Kleinberg	0.345884
Brin	0.345884
P	0.719967
Q	0.450952
R	0.820831
S	0.820831
T	0.647477
B	0.594175
J	1.148130
G	1.148130
WWW	0.445696
SIGIR	0.345166
VLDB	0.200611

Table 3.3 Alice's personal rank values

In our example, we considered only one user Alice exchanges context with. In the previous table, Table 3.4, the rankings are computed as if Alice is fully trusting Bob, and so does not make any difference between the resources she already has and the ones she receives from him. This translates into a vector e having all elements 1. If Alice doesn't trust Bob 100%, she will have to bias the PageRank on her resources, that is assign values less than 1 to the elements in the e vector corresponding to the resources coming from Bob. This means that the probability of reaching the resources she receives from Bob through a random jump is less than the probability of jumping to one of her own resources. In our experiments we computed Alice's rankings for different levels of trust she has for Bob, and the results are presented in Table 3.5.

As we would expect, the rankings decrease, as trust decreases, but are still greater than the original rankings for the resources existing on the desktop before receiving input from other users. Even for a trust level of 1%, the rankings are still greater than the ones computed for resources originally existing on one's desktop. That is because for these resources Alice already has her own ratings and they will increase due to the fact that they are referenced by some of the received resources. On the other hand, for the newly received resources, (Teevan, Dumais, etc.) for a trust level of 1%, the rankings are not much greater than 0. The probability to jump to one of these resources is 0.01, in contrast to the probability of executing a random jump to the ones already existing.

Resource	Rank	Resource	Rank
W.Y. Ma	0.524152	Teevan	0.245807
Riedl	0.369096	Rose	0.260636
Vogt	0.369096	Guha	0.260636
Cai	0.313851	Rocha	0.260636
Balmin	0.341437	Quan	0.260636
Kleinberg	0.951419	I	1.463327
Brin	0.639601	H	1.698574
P	1.351007	D	0.727691
Q	0.846576	E	0.563505
R	1.286748	F	0.622081
S	1.286748	K	0.733372
T	0.962533	L	0.650674
B	1.124971	M	0.650674
J	3.008105	N	0.650674
G	2.875325	O	0.650674
WWW	1.390752	C	0.655163
SIGIR	0.512907	A	0.406226
VLDB	0.245718	SIGCHI	0.197904
Widom	0.399090	INTERACT	0.202887
Nejdl	0.455166	ESWC	0.205699
Dumais	0.379546	DELOS	0.184534

Table 3.4 Alice's personal ranking values after receiving context information from Bob

3.4.4 Discussion

FOAF is a nice vocabulary to describe social networks, but most of the current applications are centered around describing social networks and not how to use them. The previous sections explored how to build upon FOAF and rich semantic web metadata to exchange and recommend context information and resources in a social network. These contextual metadata are described by appropriate annotation ontologies, and are exchanged within FOAF groups as specified by the group members. The exchange of metadata is done by means of additional attachments for each document exchanged via email, extending email exchange from pure document exchange to an exchange of both document and relevant context information. We presented how the computation of ranking is accomplished and how this computation is influenced by the context exchange as well as by the reputation of persons involved in the exchange process.

There are quite a few interesting issues to be investigated in future work, including privacy and security issues. This is especially important if we exploit peer-to-peer

Resource Label	**PageRank**				
	90% Trust	50% Trust	30% Trust	10% Trust	1% Trust
Wei-Ying Ma	0.512070	0.465217	0.441329	0.417406	0.406896
Riedl	0.363519	0.342122	0.331131	0.320116	0.315321
Vogt	0.363519	0.342122	0.331131	0.320116	0.315321
Cai	0.310070	0.295519	0.288060	0.280584	0.277323
Balmin	0.333760	0.303559	0.288305	0.273040	0.266257
Kleinberg	0.920420	0.799795	0.738430	0.676985	0.649917
Brin	0.622799	0.557678	0.524464	0.491199	0.476592
P	1.307479	1.137762	1.051539	0.965213	0.927123
Q	0.819327	0.712840	0.658819	0.604739	0.580836
R	1.254108	1.127853	1.063365	0.998769	0.970455
S	1.254108	1.127853	1.063365	0.998769	0.970455
T	0.940389	0.854556	0.810777	0.766931	0.747677
B	1.079916	0.902077	0.812442	0.722759	0.682798
J	2.903632	2.496173	2.289197	2.081974	1.990525
G	2.776905	2.392989	2.197998	2.002777	1.916611
WWW	1.325955	1.070835	0.942042	0.813164	0.755852
SIGIR	0.500467	0.451987	0.427339	0.402658	0.391778
VLDB	0.241880	0.226780	0.219153	0.211520	0.208128
Widom	0.371165	0.260322	0.204639	0.148938	0.124017
Nejdl	0.409618	0.227602	0.136575	0.045555	0.004607
Dumais	0.344211	0.203102	0.132490	0.061876	0.030133
Teevan	0.221220	0.122908	0.073747	0.024589	0.002470

Table 3.5 Alice's ranking values after receiving context information from Bob for different trust levels

infrastructures instead of email attachments to implement a knowledge sharing infrastructure as described in this section. It is also worthy to note that the ranking we compute for different resources can be compared to the ratings which are used in recommender systems. We can therefore not only share resources which are semantically connected to the ones we are exchanging, but also resources which are ranked / rated highly by peers in our community. An additional interesting aspect is to explore dynamic social networks, where groups are not statically defined from the beginning but dynamically based on the exchange of context metadata. In this case users can initially choose which pieces of metadata information they want to append to a document for certain recipients, and common exchange patterns then determine common interest groups and allow automatic exchange of metadata based on these previous interactions.

3.5 Personalizing Ranking over Distributed Contexts

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, in the next sections we provide a solution about how ranking can be computed in a distributed scenario, and more precisely how it can be personalized.

3.5.1 Which Information Should We Exchange?

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 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 [CGNP05]), 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.

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 ([CLC95, GIG01]). 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 [CLC95]), 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.

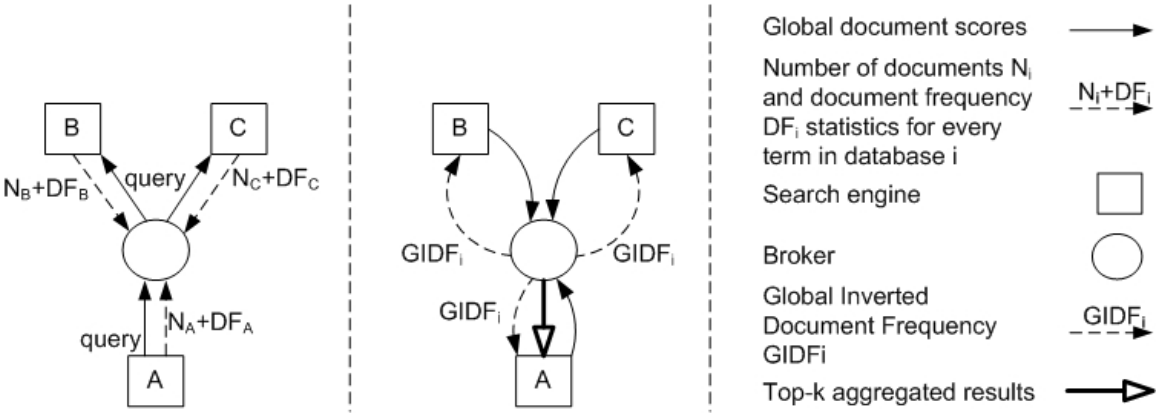

Figure 3.8 Statistics propagation for results merging

Figure 3.8 illustrates this process in detail.

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. [7]
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.

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 [CNP07]. 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

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

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:

$$R = d \cdot A \cdot R + (1 - d) \cdot e \tag{3.3}$$

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 [DNP05]. 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 [CGNP06] for a more detailed description of the algorithm).

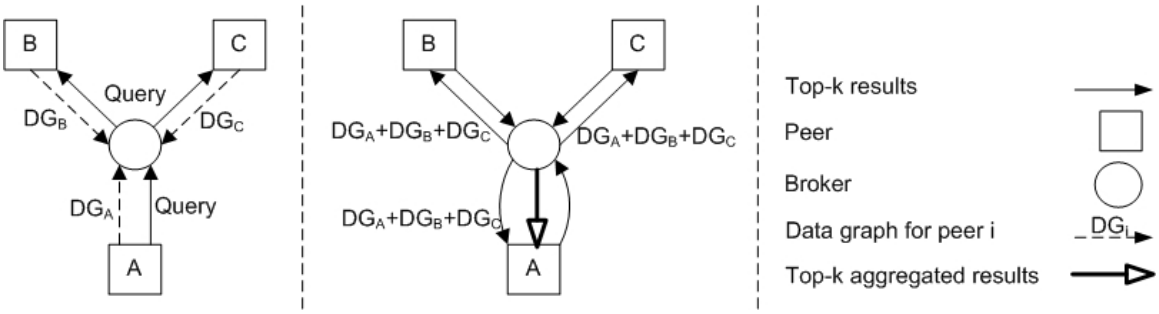

Figure 3.9 Aggregated ObjectRank computation

Let us 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 3.9 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*[8] 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.

3.5.2 Information Exchange and Rank Computation

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 [CDKS01] (which appropriately aggregates node and link information of the graph)

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

Ranking computation can be based on: a) simple ObjectRank; or b) ObjectRank with biasing [CGNP06] 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 [CGNP05]. $G'_i = (V'_i, E'_i)$ 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 $E'_i = \{e_{jk} | j, k \in V'_i, j \neq k\}$. $G_i^{anon} = (V_i^{anon}, E_i^{anon})$ 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 3.10[9].

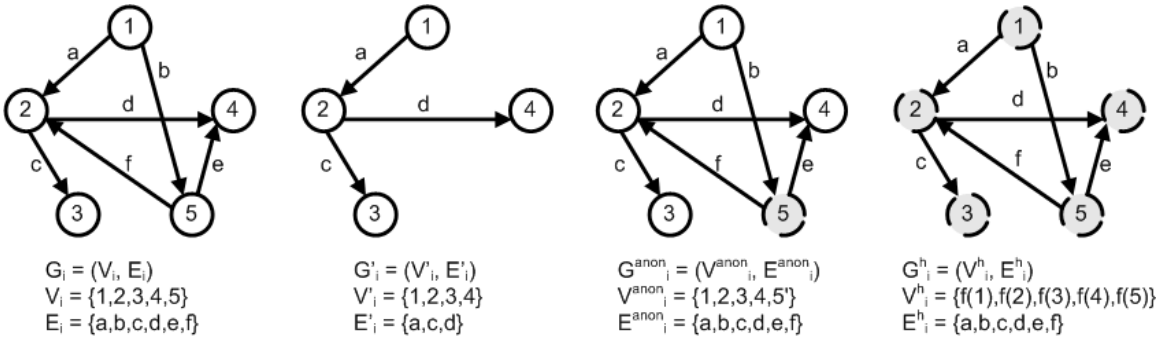

Figure 3.10 Example of weighted data graphs - different setups

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 3.11, 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 the previous section we define $G_i^{WN} = (V_i^{WN}, E_i^{WN})$, where $V_i^{WN} = WN$ and E_i^{WN} is formed as follows:

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.

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

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.

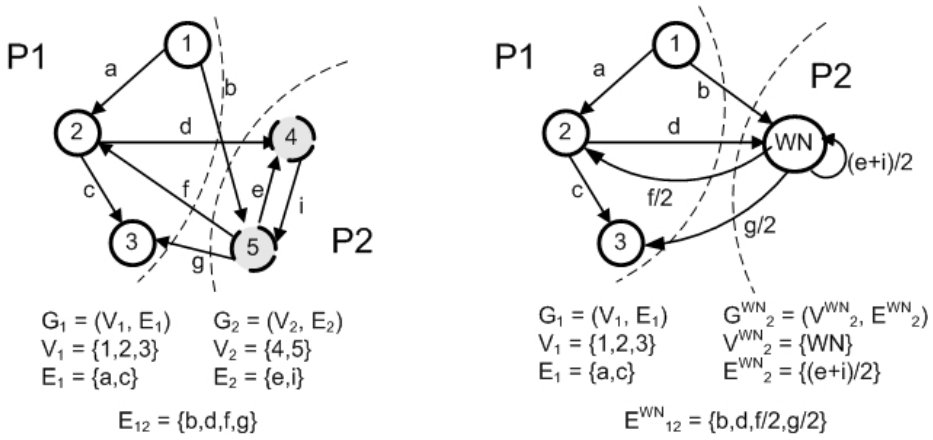

Figure 3.11 Example of world node creation

In figure 3.11 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 3.11 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.

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 3.5.2). 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.
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

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.

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.

3.5.3 Experiments

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++ [Bea06], 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 3.12 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:

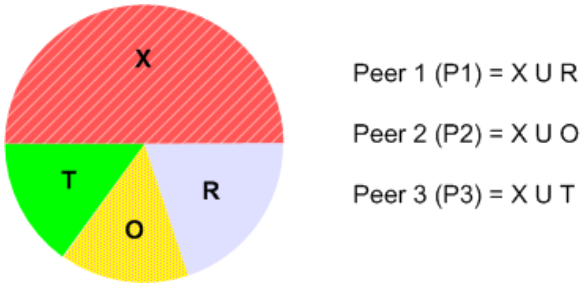

Figure 3.12 Peers' resource distribution

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.

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 3.5.2, 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 [Hav02]):

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

$$\frac{|Top_n(\tau_1) \cap Top_n(\tau_2)|}{n} \tag{3.4}$$

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:

$$KSim(\tau_1, \tau_2) = \frac{|(u,v) : \begin{matrix}\tau_1' \text{ and } \tau_2' \text{ agree on or-} \\ \text{der (u,v), and } u \neq v\end{matrix}|}{|U| \cdot |U-1|} \tag{3.5}$$

Results and Analysis

For all three setups we computed KSim and OSim measures (Tables 3.6 - 3.10), 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[10] ranked results for each algorithm.

Partitioning 1.

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

Table 3.6 SETUP 1 - OSim, KSim

In SETUP 1 (Table 3.6), 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[11]. In terms of both KSim and OSim, the second algorithm

[10] 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.

[11] 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.

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											
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 3.7 SETUP 2 - OSim, KSim

SETUP 2 (Table 3.7) 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 peer does not disclose any sensible information - the authority transfer among the peers is captured within this model.

Partitioning 2.

In SETUP 3 (Tables 3.8 - 3.10) we experimented with 3 different sizes of the 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

Table 3.8 SETUP 3 - Small 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 3.9 SETUP 3 - Medium 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

Table 3.10 SETUP 3 - Big 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.

3.5.4 Discussion

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 the previous sections 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.

4

Contributions and Open Directions

In our everyday life we are faced with high volumes of information, coming from various sources: people, mass media, World Wide Web, and even from our personal computers. It has become rather difficult to manage all this data daily. Given our every day tasks, tools for managing and organizing our data are so much needed. Present desktop search engines still rely on TFxIDF techniques, only a few keep track of metadata of resources, and even fewer employ a ranking mechanism. Users today need to organize their data according to their present tasks and working contexts, so that they can easily find and explore their needed data. And if we think of the vast amounts of data that come from the Web, the user might become againd overwhelmed, so a collaborative tool that would offer again additional metadata and a ranking mechanism would be much needed. In this thesis, we investigated first how to enhance current desktop resources with metadata which help in recreating working contexts to the user, then how to use them for providing a ranking mechanism on the desktop, and then how to compute such a ranking in a distributed, collaborative environment, where users recommend and exchange data in between their desktops. This section first summarizes all the important contributions of our work and then underlines some of the major future research directions which still remained open.

Summary of Contributions

As we have already discussed, the overwhelming data on the desktop needs some management tools which can make use of metadata capable of recreating working contexts. We focused on such metadata generation in Chapter 2, where we first show our fully implemented Beagle^{++} system, and mainly focus on the modules responsible for metadata generation. The "File Paths" module enhances each word in the path of a file with additional metadata from Wordnet - synonyms, hypernyms, hyponyms, meronyms, holonyms. The visited web links are annotated with the incoming and outgoing links in the "Web Cache" module, and the connections between an email and its attachment are recreated in the "Email Metadata" part. Our system also

has a semantic ranking module which combines a semantic ranking with a traditional TFxIDF mechanism, which we also proven to improve the search results we receive from the tool. Also, we experimented to prove the added value that our metadata generators bring, by comparing the Beagle system with our enhanced one.

Two methods for recreating contexts on the desktop were tested. The first one uses time stamps in order to determine if two files were accessed in a sequence. The main idea is that when we work for a task, we tend to access only some files and in a certain order. Therefore we constructed some algorithms which state that if two files were accessed in a sequence several times and within a small time frame, then they are related and thus a connection should be kept to reflect this. The other method builds a Bayesian Network from various evidences that it gathers for each pair of files on the desktop: textual similarity, usage activity, files opened concurrently (last two similar to the previous method) and folder hierarchy. Then the Bayesian Network is capable of detecting working contexts based on these evidences and also on a small input from the user in order to determine some contexts that the network can work with.

The last contribution of this chapter already hints to the idea that normally users don't work only with their local resources, but with others located remotely, such as web pages on the Web. The idea is that we use the personal desktop in order to better contextualize our work by providing personalized annotations for the web pages we visit. The used algorithms are grouped into three categories: document oriented where we first find similar documents on the desktop to the web page, keyword oriented where we first extract keywords from the web page and using those we find relevant keywords on the desktop, and hybrid methods, which combine the first two, by first extracting the keywords from a web page and then for each keyword we find the relevant documents on the desktop and from them we extract keywords as candidates for annotations. While measuring precision, we observed for the first method values as high as 81% for the lexical compounds method, 82% for the second method and 80% for the hybrid extraction. We also show several experiments regarding the influence of the size of the web page on the generated annoations and also show some concrete annotations for some web pages, which clearly demonstrate that our algorithms are capable of generating good quality annotations which reflect the user's interest as depicted by his desktop.

Chapter 3 is entirely dedicated to ranking, since just providing more results to a user due to the introduction of metadata is not enough. We need some ordering for these results, so that the additional metadata doesn't make it even more difficult to the user to find his needed results. The first ranking technique makes use of the metadata that we showed how to generate in the previous chapter - the time-related metadata. Wherever we have such connections, we modify the adjancency matrix and use it to compute rankings with it. Several methods were also used, depending on the way we represent the time connections - by looking at the access sequences or

at the activity sessions.

After this, we moved to the collaborative environment, where a user recommends and receives recommendations within his social group. Employing a similar semantic ranking mechanism, we investigate how the rankings change when resources are exchanged and also how the trust within the persons that send recommendations influences the way we receive these resources, in particular in the ranking values that they will have. This is reflected in the values employed in the vector describing the random jump. We experimented within a research scenario where people exchange papers and together with that their metadata: authors, conferences, years, tracks, *etc.*, all reflected within the authority transfer schemas employed. Using these schemas, we are able to model and observe the influence of the different level of trusts on the persons with which we are sharing resources.

In a distributed environment, where different desktops conform to different representational schema, we envisioned an algorithm for how people exchange resources. In this algorithm, we also took into account privacy issues, since the user is not always willing to share all his data, and not even the substructures he has on his computer. The most efficient for keeping privacy proved to be the world node method, where we comprise all the information into one single node in the resources graph, but keeping all the incoming and outgoing links, in order not to alter the ranking values. In order to fully show our cases, we also experimented with various sizes of the data exchanged, and also different sizes of the overlapping resources. This way we were able to observe that depending on these factors regarding data sizes, different algorithms were better suited. For many cases, we were able to reach values of 100% for OSim, which shows that the top resources are still all retrieved, and values of KSim also close to 100%, which shows that also the ordering of the resources is very close to the ideal case where users share everything. This is very encouraging, especially in the case of the algorithm employing a world node, since this means that we are able to share a lot of information, but not actually disclosing our internal structures and resources.

Open Directions

Since any research always opens new directions, we will next present the ones that arose from all the work presented in this thesis. For our Beagle^{++} system, we envision several improvements for each of the modules it comprises of, but in particular, for the querying part, it is obviously needed a task/context detection as the one we envisioned in the methods we presented in Sections 2.4 and 2.5. This would be needed, since it would be able to disambiguate the meaning of the user's query and push more valuable results to the user, leading to a better performance of the search engine.

For the context detection algorithms using either time stamps or a Bayesian Network, we feel the need of having more experiments which vary the influencing factors like experimenting with other clustering methods which might overcome the need of stating how many contexts the user has on his desktop. Or in the same direction for

the Bayesian Network, we would like to be able to exactly identify the contexts which don't have a special label given by the user, and apply some other technique, maybe again clustering for doing so.

When generating annotations, an obvious step is to expand this towards the collaborative environment - we can envision implementing a shared server approach that supports social tagging within folksonomies - the system would know about personal annotations from other users and would provide the most popular annotations, e.g., the ones with the highest score. This would enable the sharing of the automatic generated personal annotations in a collaborative environment, and would simply automatically create, apply and share tags dynamically.

For the desktop ranking using time links we intend to explore content based heuristics to provide us with further additional links between similar desktop documents. Also, we would like to analyze the necessity and benefits of enabling desktop search restrictions to only some specific sub-tree of the local file hierarchy. Finally, we intend to devise several methods to detect activity contexts based on activity analysis, and then integrate them into the ranking scheme itself.

There are quite a few interesting issues to be investigated as future work for the recommendation mechanisms, including privacy and security issues. This is especially important if we exploit peer-to-peer infrastructures instead of email attachments to implement a knowledge sharing infrastructure as described in this thesis. It is also worthy to note that the ranking we compute for different resources can be compared to the ratings which are used in recommender systems. We can therefore not only share resources which are semantically connected to the ones we are exchanging, but also resources which are ranked / rated highly by peers in our community. An additional interesting aspect is to explore dynamic social networks, where groups are not statically defined from the beginning but dynamically based on the exchange of context metadata. In this case users can initially choose which pieces of metadata information they want to append to a document for certain recipients, and common exchange patterns then determine common interest groups and allow automatic exchange of metadata based on these previous interactions. Also, for the distributed enviroment, 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.

A

Curriculum Vitae

Stefania Costache, born on December 27^{th} 1980, in Buzau, Romania.

Oct. 2004 -	Junior researcher and Ph.D student at Forschungszentrum L3S, Universität Hannover
Apr. - July. 2004	Master Studies in Computer Science, Ecole Supérieure d'Electricité, SUPELEC, Paris, France Title of the thesis: *"Hypermédias Adaptatifs"*
1999 - 2004	Bachelor Studies in Computer Science, Politehnica University, Bucharest, Romania
Jun. 2001 - Feb. 2003	C++ Developer, Crystal Interactive Systems, Bucharest, Romania www.crystalinter.com
Jul. - Aug. 2001	Practice at Policolor, Bucharest, Romania www.policolor.ro

Bibliography

[AKM+03] Harith Alani, Sanghee Kim, David E. Millard, Mark J. Weal, Wendy Hall, Paul H. Lewis, and Nigel R. Shadbolt. Automatic ontology-based knowledge extraction from web documents. *IEEE Intelligent Systems*, 18(1):14–21, 2003.

[AMHAS03] Boanerges Aleman-Meza, Chris Halaschek, I. Budak Arpinar, and Amit Sheth. Context-aware semantic association ranking. In *Semantic Web and Databases Workshop*, 2003.

[AT99] Peter G. Anick and Suresh Tipirneni. The paraphrase search assistant: Terminological feedback for iterative information seeking. In *Proc. of the 22nd Annual Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1999.

[BCC+06] I. Brunkhorst, P. A. Chirita, S. Costache, J. Gaugaz, E. Ioannou, T. Iofciu, E. Minack, W. Nejdl, and R. Paiu. The Beagle++ Toolbox: Towards an Extendable Desktop Search Architecture. In *Proceedings of Semantic Desktop and Social Semantic Collaboration Workshop, ISWC*, 2006.

[BCN06] Jürgen Belizki, Stefania Costache, and Wolfgang Nejdl. Application independent metadata generation. In *CAMA '06: Proceedings of the 1st international workshop on Contextualized attention metadata: collecting, managing and exploiting of rich usage information*, pages 33–36, New York, NY, USA, 2006. ACM.

[Bea06] Beagle++. http://beagle.kbs.uni-hannover.de/, 2006.

[BGS05] Monica Bianchini, Marco Gori, and Franco Scarselli. Inside pagerank. *ACM Trans. Inter. Tech.*, 5(1):92–128, 2005.

[BGSV06] Stephan Bloehdorn, Olaf Görlitz, Simon Schenk, and Max Völkel. TagFS - Tag Semantics for Hierarchical File Systems. In *I-KNOW*, 2006.

[BH99] Jay Budzik and Kristian Hammond. Watson: Anticipating and contextualizing information needs. In *Proceedings of the Sixty-second Annual Meeting of the American Society for Information Science*, 1999.

[BHB01] Jay Budzik, Kristian J. Hammond, and Lawrence Birnbaum. Information access in context. *Knowl.-Based Syst.*, 14:37–53, 2001.

[BHP04] A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB*, Toronto, September 2004.

[BKvH02] J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *ISWC*, 2002.

[BM06] Christopher H. Brooks and Nancy Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *Proc. of the 15th World Wide Web Conference*, 2006.

[BN95] Deborah Barreau and Bonnie Nardi. Finding and reminding: File organization from the desktop. *ACM SIGCHI Bulletin*, 27(3):39–43, 1995.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[CCGN06] P.-A. Chirita, S. Costache, J. Gaugaz, and W. Nejdl. Desktop context detection using implicit feedback. In *PIM*, 2006.

[CCNH07] Paul A. Chirita, Stefania Costache, Wolfgang Nejdl, and Siegfried Handschuh. P-tag: large scale automatic generation of personalized annotation tags for the web. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 845–854, New York, NY, USA, 2007. ACM.

[CDKS01] S. Chien, C. Dwork, S. Kumar, and D. Sivakumar. Towards exploiting link evolution. In *Unpublished manuscript.*, 2001.

[CFN06a] Paul Alexandru Chirita, Claudiu Firan, and Wolfgang Nejdl. Summarizing local context to personalize global web search. In *Proc. of the 15th Intl. CIKM Conf. on Information and Knowledge Management*, 2006.

[CFN06b] Paul Alexandru Chirita, Claudiu S. Firan, and Wolfgang Nejdl. Pushing task relevant web links down to the desktop. In *Proc. of the 8th ACM Intl. Workshop on Web Information and Data Management held at the 15th Intl. ACM CIKM Conference on Information and Knowledge Management*, 2006.

[CGG+05] Paul Alexandru Chirita, Rita Gavriloaie, Stefania Ghita, Wolfgang Nejdl, and Raluca Paiu. Activity based metadata for semantic desktop search. In *Proc. of the 2nd European Semantic Web Conference*, Heraklion, Greece, May 2005.

[CGIN10] S. Costache, J. Gaugaz, E. Ioannou, and W. Nejdl. Detecting contexts on the desktop using bayesian networks. In *Desktop Search Workshop*, 2010.

[CGNP05] Paul Alexandru Chirita, Stefania Ghita, Wolfgang Nejdl, and Raluca Paiu. Semantically enhanced searching and ranking on the desktop. In *Proc. of the Semantic Desktop Workshop held at the 4th International Semantic Web Conference*, 2005.

[CGNP06] Paul Alexandru Chirita, Stefania Ghita, Wolfgang Nejdl, and Raluca Paiu. Beagle++: Semantically enhanced searching and ranking on the desktop. In *Proc. of the 3rd European Semantic Web Conf.*, 2006.

[CH98] Chia-Hui Chang and Ching-Chi Hsu. Integrating query expansion and conceptual relevance feedback for personalized web information retrieval. In *Proc. of the 7th Intl. Conf. on World Wide Web*, 1998.

[CHBG01] Leslie Carr, Wendy Hall, Sean Bechhofer, and Carole Goble. Conceptual linking: ontology-based open hypermedia. In *Proc. of the 10th Intl. Conf. on World Wide Web*, 2001.

[CHS04] P. Cimiano, S. Handschuh, and S. Staab. Towards the self-annotating web. In *Proceedings of the 13th World Wide Web Conference*, 2004.

[CHSS08] Chaitanya Chemudugunta, America Holloway, Padhraic Smyth, and Mark Steyvers. Modeling Documents by Combining Semantic Concepts with Unsupervised Statistical Learning. In *ISWC*, 2008.

[CLC95] J. P. Callan, Z. Lu, and W. Bruce Croft. Searching distributed collections with inference networks. In *Proc. of the Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, 1995.

[CLS05] Philipp Cimiano, Günter Ladwig, and Steffen Staab. Gimme' the context: context-driven automatic semantic annotation with c-pankow. In *Proc. of the 14th World Wide Web Conference*, 2005.

[CLWB01] Mark Claypool, Phong Le, Makoto Wased, and David Brown. Implicit interest indicators. In *Proc. of the 6th ACM IUI Intl. Conf. on Intelligent User Interfaces*, 2001.

[CMBT02] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. Gate: A framework and graphical development environment for robust nlp tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, 2002.

[CNP07] Stefania Costache, Wolfgang Nejdl, and Raluca Paiu. Personalizing pagerank-based ranking over distributed collections. In *CAiSE*, pages 111–126, 2007.

[CPG05] Adam Cheyer, Jack Park, and Richard Giuli. IRIS: Integrate. Relate. Infer. Share. In *SemDeskWS*, 2005.

[CPKT92] Douglas R. Cutting, Jan O. Pedersen, David R. Karger, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *SIGIR*, 1992.

[DCC+03] S. Dumais, E. Cutrell, JJ Cadiz, G. Jancke, R. Sarin, and Daniel C. Robbins. Stuff i've seen: A system for personal information retrieval and re-use. In *SIGIR*, 2003.

[DDL+90] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[DEG+03] S. Dill, N. Eiron, D. Gibson, D. Gruhl, and R. Guha. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th World Wide Web Conference*, 2003.

[DF04] Stefan Decker and Martin Frank. The social semantic desktop. In *DERI Technical Report 2004-05-02*, 2004.

[DFJ+04] 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 Conference on Information and Knowledge Management*, 2004.

[DH05] X. Dong and A. Y. Halevy:. A platform for personal information management and integration. In *Proc. of Conf. on Innovative Data Systems Research (CIDR)*, 2005.

[DHN+04] X. Dong, A. Y. Halevy, E. Nemes, S. B. Sigundsson, and P. Domingos. SEMEX: Toward On-the-Fly Personal Information Integration. In *IIWeb*, 2004.

[DK04] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. In *ACM Transactions on Information Systems*, January 2004.

[DNP05] Andrei Damian, Wolfgang Nejdl, and Raluca Paiu. Peer-sensitive objectrank: Valuing contextual information in social networks. In *Proc. of the International Conference on Web Information Systems Engineering*, November 2005.

[Dun93] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19:61–74, 1993.

[Edm69] H. P. Edmundson. New methods in automatic extracting. *Journal of the ACM*, 16(2):264–285, 1969.

[Eft95] Efthimis N. Efthimiadis. User choices: A new yardstick for the evaluation of ranking algorithms for interactive query expansion. *Information Processing and Management*, 31(4):605–620, 1995.

[Fal04] 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.

[FF95] E. Freeman and S. Fertig. Lifestreams: Organizing your electronic life. In *Proc. of the AAAI Symposium on AI Applications in Knowledge Navigation and Retrieval*, 1995.

[FHM05] M. Franklin, A. Y. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Rec.*, 34(4):27–33, 2005.

[FKM+03] Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. Evaluating implicit measures to improve the search experience. In *Workshop on Implicit Measures of User Interests and Preferences, SIGIR 2003*, 2003.

[FMNW03] Dennis Fetterly, Mark Manasse, Marc Najork, and Janet Wiener. A large-scale study of the evolution of web pages. In *Proc. of the 12th Intl. Conf. on World Wide Web*, 2003.

[GBL+02] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. Mylifebits: fulfilling the memex vision. In *Proc. of the ACM Conference on Multimedia*, 2002.

[GCC+08] Julien Gaugaz, Stefania Costache, Paul Alexandru Chirita, Claudiu Firan, and Wolfgang Nejdl. Activity based links as a ranking factor in semantic desktop search. In *LA-WEB*, 2008.

[Ghi05] Stefania Ghita. Using your desktop as personal digital library, 2005.

[GHM+07] Tudor Groza, Siegfried Handschuh, Knud Moeller, Gunnar Grimnes, Leo Sauermann, Enrico Minack, Cedric Mesnage, Mehdi Jazayeri, Gerald Reif, and Rosa Gudjonsdottir. The NEPOMUK Project – On the way to the Social Semantic Desktop. In *I-SEMANTICS*, 2007.

[GIG01] Noah Green, Panagiotis G. Ipeirotis, and Luis Gravano. SDLIP + STARTS = SDARTS a protocol and toolkit for metasearching. In *ACM/IEEE Joint Conference on Digital Libraries*, pages 207–214, 2001.

[GMM03] 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.

[GNOT92] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collabarative filtering to weave an information tapestry. In *ACM Press*, December 1992.

[GNP05a] 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.

[GNP05b] Stefania Ghita, Wolfgang Nejdl, and Raluca Paiu. Semantically rich recommendations in social networks for sharing and exchanging semantic context. In *In ESWC Workshop on Ontologies in P2P Communities*, 2005.

[GSS06] Olaf Görlitz, Simon Schenk, and Steffen Staab. TagFs - Bringing Semantic Metadata to the Filesystem. Demo at ESWC, 2006.

[GWR99] Susan Gauch, Jianying Wang, and Satya Mahesh Rachakonda. A corpus analysis approach for automatic query expansion and its extension to multiple databases. *ACM Transactions on Information Systems*, 17(3):250–250, 1999.

[Hav02] T. Haveliwala. Topic-sensitive pagerank. In *In Proceedings of the Eleventh International World Wide Web Conference, Honolulu, Hawaii*, May 2002.

[Hea99] Marti A. Hearst. Untangling text data mining. In *Proc. of the 37th Meeting of the Association for Computational Linguistics on Computational Linguistics*, 1999.

[HKQ02] D. Huynh, D. Karger, and D. Quan. Haystack: A platform for creating, organizing and visualizing information using rdf. In *Proc. of the Sem. Web Workshop held at 11th World Wide Web Conf.*, 2002.

[HP96] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *SIGIR*, 1996.

[HS02] S. Handschuh and S. Staab. Authoring and annotation of web pages in cream. In *Proc. of the 11th Intl. World Wide Web Conf.*, 2002.

[JDB02] William Jones, Susan Dumais, and Harry Bruce. Once found, what then?: A study of keeping behaviors in the personal use of web information. In *Proc. of ASIST*, 2002.

[KBH+03] David R. Karger, Karum Bakshi, David Huynh, Dennis Quan, and Vineet Sinha. Haystack: A customizable general-purpose information management tool for end users of semistructured data. In *Proc. of the 1st Intl. Conf. on Innovative Data Syst.*, 2003.

[KC99] M-C. Kim and K. Choi. A comparison of collocation based similarity measures in query expansion. *Information Processing and Management*, 35:19–30, 1999.

[KKPS01] J. Kahan, M. Koivunen, E. Prud'Hommeaux, and R. Swick. Annotea: An Open RDF Infrastructure for Shared Web Annotations. In *Proc. of the 10th Intl. World Wide Web Conf.*, 2001.

[KMM+97] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 40:77–87, 1997.

[KOR00] J. Kim, D. Oard, and K. Romanik. User modeling for information access based on implicit feedback. In *Technical Report*, 2000.

[KPO+03] Atanas Kiryakov, Borislav Popov, Damyan Ognyanoff, Dimitar Manov, Angel Kirilov, and Miroslav Goranov. Semantic annotation, indexing, and retrieval. In *International Semantic Web Conference*, 2003.

[KT03] Diane Kelly and Jaime Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37:18–28, 2003.

[LAJ01] Adenike M. Lam-Adesina and Gareth J. F. Jones. Applying summarization techniques for term selection in relevance feedback. In *Proc. of the 24th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2001.

[LGZ08] Xin Li, Lei Guo, and Yihong Eric Zhao. Tag-based Social Interest Discovery. In *WWW*, 2008.

[Mal83] T. Malone. How do people organize their desks? implications for the design of office information systems. *ACM Transactions on Office Information Systems*, 1(1):99–112, 1983.

[MBC03] D. Maynard, K. Bontcheva, and H. Cunningham. Towards a semantic extraction of named entities. In *Recent Advances in Natural Language Processing*, 2003.

[MdRA+08] Fernando Mourão, Leonardo C. da Rocha, Renata Braga Araújo, Thierson Couto, Marcos André Gonçalves, and Wagner Meira. Understanding temporal aspects in document classification. In *WSDM*, 2008.

[MGHN05] Integrating Contextual Metadata, Stefania Ghita, Nicola Henze, and Wolfgang Nejdl. Task specific semantic views: Extracting and. In *In Submitted for publication, L3S Technical Report*, 2005.

[Mil95] G.A. Miller. Wordnet: An electronic lexical database. *Communications of the ACM*, 38(11):39–41, 1995.

[MKR04] Bradley N. Miller, Joseph A. Konstan, and John Riedl. Pocketlens: Toward a personal recommender system. *ACM Trans. Inf. Syst.*, 22(3):437–476, 2004.

[MPC+10] Enrico Minack, Raluca Paiu, Stefania Costache, Gianluca Demartini, Julien Gaugaz, Ekaterini Ioannou, Paul-Alexandru Chirita, and Wolfgang Nejdl. Leveraging personal metadata for desktop search: The beagle++ system. *J. Web Sem.*, 8(1):37–54, 2010.

[MS94] Masahiro Morita and Yoichi Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, 1994.

[MSG+08] Enrico Minack, Leo Sauermann, Gunnar Grimnes, Christiaan Fluit, and Jeen Broekstra. The Sesame LuceneSail: RDF Queries with Fulltext Search. Technical report, NEPOMUK 2008-1, 2008.

[MSN09] E. Minack, W. Siberski, and W. Nejdl. Benchmarking Fulltext Search Performance of RDF Stores. In *ESWC*, 2009.

[NEP] NEPOMUK. The social semantic desktop. http://nepomuk.semanticdesktop.org/.

[NHCS07] David Newman, Kat Hagedorn, Chaitanya Chemudugunta, and Padhraic Smyth. Subject Metadata Enrichment using Statistical Topic Models. In *JCDL*, 2007.

[NHW+04] 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.

[OK01] D. W. Oard and J. Kim. Modeling information content using observable behavior. In *Proceedings of the 64th Annual Meeting of the American Society for Information Science and Technology*, 2001.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[PDMW06] J. X Parreira, D. Donato, S. Michel, and G. Weikum. 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.

[Pea88] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[QK04] Dennis Quan and David Karger. How to make a semantic web browser. In *Proceedings of the 13th International WWW Conference*, 2004.

[RIS+94] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM Press, 1994.

[RMO+93] Daniel Rose, Richard Mander, Tim Oren, Dulce Ponceleon, Gitta Salomon, and Yin Wong. Content awareness in a file system interface: Implementing the 'pile' metaphor for organizing information. In *Proc. of the 16th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1993.

[Roc71] J. Rocchio. Relevance feedback in information retrieval. *The Smart Retrieval System: Experiments in Automatic Document Processing*, pages 313–323, 1971.

[RSdA04] 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.

[Sau03] Leopold Sauermann. Using semantic web technologies to build a semantic desktop. Master's thesis, TU Vienna, 2003.

[SCDN10a] A. Stewart, S. Costache, K. Denecke, and W. Nejdl. Cross-corpora analysis for epidemic intelligence in blogs. In *Submitted to the 1st ACM International Health Informatics Symposium, IHI*, 2010.

[SCDN10b] A. Stewart, S. Costache, K. Denecke, and W. Nejdl. Exploiting the language of moderated sources for cross-classification of user generated content. In *Submitted to the Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2010.

[SCNN10] R. Stecher, S. Costache, C. Niederee, and W. Nejdl. Query ranking in information integration. In *Proc. of the 22nd Intl. Conf. on Advanced Information Systems Engineering, CAiSE*, 2010.

[SG05] Craig Soules and Gregory Ganger. Connections: using context to enhance file search. In *SOSP*, 2005.

[SK05] Vineet Sinha and David R. Karger. Magnet: supporting navigation in semistructured data environments. In *Proc. of the 2005 ACM SIGMOD Intl. Conf. on Management of Data*, 2005.

[SS04] Leo Sauermann and Sven Schwarz. Introducing the Gnowsis Semantic Desktop. In *Poster at ISWC*, 2004.

[SSS03] N. Stojanovic, R. Studer, and L. Stojanovic. An approach for the ranking of query results in the semantic web. In *ISWC*, 2003.

[Sta97] Mark A. Stairmand. Textual context analysis for information retrieval. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, 1997.

[TAAK04] J. Teevan, C. Alvarado, M.S. Ackerman, and D. Karger. The Perfect Search Engine Is Not Enough: A Study of Orienteering Behavior in Directed Search. In *Proc. of CHI*, 2004.

[TCRS07] Thanh Tran, Philipp Cimiano, Sebastian Rudolph, and Rudi Studer. Ontology-Based Interpretation of Keywords for Semantic Search. In *ISWC*, 2007.

[The] The foaf project. http://www.foaf-project.org/.

[VWMFC05] Vishwa Vinay, Ken Wood, Natasa Milic-Frayling, and Ingemar J. Cox. Comparing relevance feedback algorithms for web search. In *Proc. of the 14th Intl. Conf. on World Wide Web*, 2005.

[WA04] Jie Wu and Karl Aberer. Using SiteRank for P2P Web Retrieval, 2004.

[WD04] Yuan Wang and David J. DeWitt. Computing PageRank in a distributed internet search system. In *Proceedings of the 30th VLDB Conference*, 2004.

[WT06] Shao-Chi Wang and Yuzuru Tanaka. Topic-oriented query expansion for web search. In *Proc. of the 15th Intl. Conf. on World Wide Web*, pages 1029–1030, 2006.

[WZ02] Yimin Wu and Aidong Zhang. Category-based search using metadatabase in image retrieval. In *IEEE International Conference on Multimedia and Expo*, 2002.

[WZL+08] Haofen Wang, Kang Zhang, Qiaoling Liu, Thanh Tran, and Yong Yu. Q2Semantic: A Lightweight Keyword Interface to Semantic Search. In *ESWC*, pages 584–598, 2008.

[XC96] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proc. of the 19th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 4–11, 1996.

[YP97] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, 1997.

[YSLH03] 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.

[ZWX+07] Qi Zhou, Chong Wang, Miao Xiong, Haofen Wang, and Yong Yu. SPARK: Adapting Keyword Query to Semantic Search. In *ISWC*, 2007.