Emerging Applications of Link Analysis for Ranking

Von der Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades eines
Doktor-Ingenieurs
(abgekürzt: Dr.-Ing.)
genehmigte Dissertation

Von Dipl.-Ing. Paul - Alexandru Chirita
geboren am 07.07.1980 in Bukarest, Rumänien.

2007

Kommission:

Referent:	**Prof. Dr. Wolfgang Nejdl** Gottfried Wilhelm Leibniz Universität Hannover, Hannover, Deutschland
Korreferent:	**Prof. Dr. Ricardo Baeza - Yates** Universität Pompeu - Fabra, Barcelona, Spanien
Korreferent:	**Prof. Dr. Klaus Jobmann** Gottfried Wilhelm Leibniz Universität Hannover, Hannover, Deutschland
Tag der Promotion:	24. Mai 2007

Zusammenfassung

Der starke Zuwachs von elektronisch verfügbaren Daten haben stark zur Popularität von Suchmaschinen beigetragen. Allerdings sind die Nutzer von Suchmaschinen typischerweise nur an den wenigen Dokumenten interessiert, die im Bezug auf ihre Arbeit die höchste Relevanz besitzen. Es ist also sehr wichtig hochwertige Rankingmethoden zu entwickeln, die effizient diese relevanten Dokumente für die verschiedenen Aktivitäten zur Informationssuche identifizieren, die solche Nutzer entwickeln.

Diese Arbeit enthält zwei Beiträge zu dem Bereich "Information Retrieval". Erstens identifizieren wir die Anwendungsbereiche, in den ein nutzerorientiertes Ranking derzeit nicht vorhanden ist, obwohl es extrem notwendig ist, um einen hochqualitativen Zugang zu den für einen Nutzer relevanten Ressourcen zu ermöglichen. Zweitens entwickeln wir für jeden von diesen Anwendungsbereichen die entsprechenden Rankingalgorithmen, die auf sozialen Charakteristika aufbauen und diese ausnutzen, entweder auf einem makroskopischen oder einem mikroskopischen Niveau. Dies wird durch "Link Analysis" Techniken erreicht, die auf der graphbasierten Darstellung der Verknüpfungen zwischen Objekten bauen, um sie zu ordnen oder einfach um Muster im Bezug auf deren soziale Eigenschaften zu erkennen.

Wir fangen an und argumentieren, dass das Ranken von Objekten auf dem Desktop sehr effektiv den Zugang zu allen Ressourcen auf dem Desktop verbessern kann. Dafür schlagen wir vor, die "Link Analysis" Methoden auch auf dem Desktop zu nutzen unter Verwendung von Statistiken ber das Nutzerverhalten. Wir zeigen, dass ein auf diese Weise entwickeltes Ranking sehr vorteilhaft für das Anwendungsszenario einer Desktop-Suchmaschine ist.

Anschlieend setzen wir dieselben grundlegenden Ideen für die Erkennung von "Spam Emails" ein. Dazu verbinden wir Menschen in sozialen Netzwerken, basierend auf dem Austausch von Emails zwischen diesen, und leiten daraus eine Reputationsmetrik ab, die böswillige Mitglieder jeder Community isoliert. Auf eine ähnliche Weise modellieren wir mehrere künstliche Linkstrukturen auf einer höheren Abstraktionsebene, die Link Analysis Algorithmen im allgemeinen negativ beeinflussen können. Wir geben auch an, wie man solche Linkstrukturen im Anwendungsszenario "Ranken von Webseiten" entfernen kann.

Der letzte Teil dieser Arbeit nutzt manuell erstellte Informationsrepositorien, um die Web Suche zu personalisieren. Wir untersuchen zwei verschiedene Arten von solchen Repositorien, solche die global bearbeitet werden können und solche die individuell bearbeitet werden können. Im ersten Fall wenden wir Link Analysis Techniken auf öffentliche Webverzeichnissen an, wie zum Beispiel das Open Directory, und definieren geeignete Ähnlichkeitsmetriken, die die Suchergebnisse nach den Präferenzen des Nutzers anordnen. Für individuell bearbeitbare Repositorien, schlagen wir eine Methode zur Erweiterung von Suchanfragen vor, die sowohl auf der Analyse von Text, als auch auf Link Analysis Methoden in Zusammenhang mit "Personal Information Repositories" beruhen. Ausführliche Experimente, die beide Vorgehensweisen auswerten, zeigen in beiden Fallen wesentliche Verbesserungen im Vergleich zu einer herkömmlichen Suche mit Google.

Schlagwörter

Informationswiedergewinnung, Data Mining, Internet

Abstract

The booming growth of digitally available information has thoroughly increased the popularity of search engine technology over the past years. At the same time, upon interacting with this overwhelming quantity of data, people usually inspect only the very few most relevant items for their task. It is thus very important to utilize high quality ranking measures which efficiently identify these items under the various information retrieval activities we pursue.

In this thesis we provide a twofold contribution to the Information Retrieval field. First, we identify those application areas in which a user oriented ranking is missing, though extremely necessary in order to facilitate a qualitative access to relevant resources. Second, for each of these areas we propose appropriate ranking algorithms which exploit their underlying social characteristics, either at the macroscopic, or at the microscopic level. We achieve this by utilizing link analysis techniques, which build on top of the graph based representation of relations between resources in order to rank them or simply to identify social patterns relative to the investigated data set.

We start by arguing that Ranking Desktop Items is very effective in improving resource access within Personal Information Repositories. Thus, we propose to move link analysis methods down to the PC Desktop by exploiting usage analysis statistics, and show the resulted importance ordering to be highly beneficial for the particular scenario of Desktop Search.

We then apply the same technique for Spam Detection. We connect people across email social networks based on their email exchanges and induce a reputation metric which nicely isolates malicious members of a community. Similarly, we model several higher level artificial constructs which could negatively manipulate generic link analysis ranking algorithms, and indicate how to remove them in the case of Web page ranking.

Finally, we exploit manually created large scale information repositories in order to Personalize Web Search. We investigate two different types of such repositories, namely globally edited ones and individually edited ones. For the former category we project link analysis onto public taxonomies such as the Open Directory and define appropriate similarity measures which order the search output in accordance to each user's preferences. For the latter one, we propose to expand Web queries by utilizing both text and link analysis on top of Personal Information Repositories. Extensive experiments analyzing both approaches show them to yield significant improvements over regular Google search.

Keywords

Information Retrieval, Data Mining, World Wide Web

Foreword

The algorithms presented in this thesis have been published within several Information Systems conferences, as follows.

The usage analysis based Desktop ranking ideas were split across two interest areas: (1) Semantic Web, when we aimed for specific user actions, modeled usually using ontologies, [61, 62, 63], and (2) Information Retrieval, when all activities were logged and analyzed from a statistical point of view [66]:

- *Beagle++: Semantically Enhanced Searching and Ranking on the Desktop.* By Paul - Alexandru Chirita, Stefania Ghita, Wolfgang Nejdl, Raluca Paiu. In Proceedings of the 3rd European Semantic Web Conference (ESWC), Budva, Montenegro, 2006 [63].
- *Activity-Based Metadata for Semantic Desktop Search.* By Paul - Alexandru Chirita, Stefania Ghita, Rita Gavriloaie, Wolfgang Nejdl, Raluca Paiu. In Proceedings of the 2nd European Semantic Web Conference (ESWC), Heraklion, Greece, 2005 [61].
- *Semantically Enhanced Searching and Ranking on the Desktop.* By Paul - Alexandru Chirita, Stefania Ghita, Wolfgang Nejdl, Raluca Paiu. In Proceedings of the Semantic Desktop Workshop held at the 3rd International Semantic Web Conference, Galway, Ireland, 2005 [62].
- *Analyzing User Behavior to Rank Desktop Items.* By Paul - Alexandru Chirita, Wolfgang Nejdl. In Proceedings of the 13th International Symposium on String Processing and Information Retrieval (SPIRE), Glasgow, United Kingdom, 2006 [66].

The other two chapters have been focused exclusively on Information Retrieval techniques. The work on spam detection was presented in less, but more important conferences, after major parts of the research had been already completed [58, 44, 28]:

- *MailRank: Using Ranking for Spam Detection.* By Paul - Alexandru Chirita, Jrg Diederich, Wolfgang Nejdl. In Proceedings of the 14th ACM International CIKM Conference on Information and Knowledge Management, Bremen, Germany, 2005 [58].
- *Site Level Noise Removal for Search Engines.* By Andre Carvalho, Paul - Alexandru Chirita, Edleno Silva de Moura, Pavel Calado, Wolfgang Nejdl. In Proceedings of the 15th International World Wide Web Conference (WWW), Edinburgh, United Kingdom, 2006 [44].
- *An Analysis of Factors used in Search Engine Ranking.* By Albert Bifet, Carlos Castillo, Paul - Alexandru Chirita, Ingmar Weber. In Proceedings of the Adversarial Information Retrieval Workshop held at the 14th International World Wide Web Conference, Chiba, Japan, 2006 [28].

The most important chapter addresses the topic Web search personalization, and is built on top of the following publications [67, 60, 59, 56]:

- *Using ODP Metadata to Personalize Search.* By Paul - Alexandru Chirita, Wolfgang Nejdl, Raluca Paiu, Christian Kohlschtter. In Proceedings of the 28th ACM International SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, 2005 [67].
- *Summarizing Local Context to Personalize Global Web Search.* By Paul - Alexandru Chirita, Claudiu Firan, Wolfgang Nejdl. In Proceedings of the 15th ACM International CIKM Conference on Information and Knowledge Management, Arlington, United States, 2006 [60].
- *P-TAG: Large Scale Automatic Generation of Personalized Annotation TAGs for the Web.* By Paul - Alexandru Chirita, Stefania Costache, Siegfried Handschuh, Wolfgang Nejdl. In Proceedings of the 16th International World Wide Web Conference (WWW), Banff, Canada, 2007 [56].
- *Pushing Task Relevant Web Links down to the Desktop.* By Paul - Alexandru Chirita, Claudiu Firan, Wolfgang Nejdl. In Proceedings of the 8th ACM Workshop on Web Information and Data Management (WIDM) held at the 15th ACM International CIKM Conference on Information and Knowledge Management, Arlington, United States, 2006 [59].

During the Ph.D. work, I have also published a number of "exercise papers", in which I mostly intended to capture the opinion of the research community upon either one of the three above mentioned topics, or a fourth application of link analysis for ranking, namely Peer-To-Peer ranking. However, in order to keep the quality of the thesis at a high level, I decided to regard these articles as "related work", and discuss them only briefly within the appropriate background sections. Here is a complete list with all of them:

- *The Beagle++ Toolbox: Towards an Extendable Desktop Search Architecture.* By Ingo Brunkhorst, Paul-Alexandru Chirita, Stefania Costache, Julien Gaugaz, Ekaterini Ioannou, Tereza Iofciu, Enrico Minack, Wolfgang Nejdl, Raluca Paiu. In Proceedings of the 2nd Semantic Desktop Workshop held at the 5th International Semantic Web Conference, Athens, United States, 2006 [37].
- *Desktop Context Detection Using Implicit Feedback.* By Paul - Alexandru Chirita, Julien Gaugaz, Stefania Costache, Wolfgang Nejdl. In Proceedings of the Workshop on Personal Information Management held at the 29th ACM International SIGIR Conf. on Research and Development in Information Retrieval, Seattle, United States, 2006. [55].
- *Building a Desktop Search Test-bed* (poster). Sergey Chernov, Pavel Serdyukov, Paul - Alexandru Chirita, Gianluca Demartini, and Wolfgang Nejdl. In Proceedings of the 29th European Conference on Information Retrieval (ECIR), Rome, Italy, 2007 [52].
- *Preventing Shilling Attacks in Online Recommender Systems.* By Paul - Alexandru Chirita, Wolfgang Nejdl, Cristian Zamfir. In Proceedings of the 7th ACM Workshop on Web Information and Data Management (WIDM) held at the 14th ACM International CIKM Conference on Information and Knowledge Management, Bremen, Germany, 2005 [70].
- *Efficient Parallel Computation of PageRank.* By Christian Kohlschtter, Paul - Alexandru Chirita, Wolfgang Nejdl. In Proceedings of the 28th European Conference on Information Retrieval (ECIR), London, United Kingdom, 2006 [145].
- *PROS: A Personalized Ranking Platform for Web Search.* By Paul - Alexandru Chirita, Daniel Olmedilla, Wolfgang Nejdl. In Proceedings of the 3rd International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AHA), Eindhoven, Netherlands, 2004 [73].
- *Finding Related Hubs on the Link Structure of the WWW.* By Paul - Alexandru Chirita, Daniel Olmedilla, Wolfgang Nejdl. In Proceedings of the 3rd IEEE / WIC / ACM International Conference on Web Intelligence (WI), Beijing, China, 2004 [72].
- *Finding Related Hubs and Authorities* (poster). By Paul - Alexandru Chirita, Daniel Olmedilla, Wolfgang Nejdl. In Proceedings of the 1st IEEE Latin-American Web (LA-Web) Congress, Santiago, Chile [71].
- *Using Link Analysis to Identify Aspects in Faceted Web Search.* By Christian Kohlschtter, Paul - Alexandru Chirita, Wolfgang Nejdl. In Proc. of the Faceted Search Workshop held at the 29th Intl. ACM SIGIR Conf. on Res. and Development in Information Retrieval, Seattle, U.S.A., 2006 [144].

- *Search Strategies for Scientific Collaboration Networks.* By Paul - Alexandru Chirita, Andrei Damian, Wolfgang Nejdl, Wolf Siberski. In Proceedings of the 2nd P2P Information Retrieval Workshop held at the 14th ACM International CIKM Conference on Information and Knowledge Management, Bremen, Germany, 2005 [57].
- *Designing Publish/Subscribe Networks using Super-Peers.* By Paul - Alexandru Chirita, Stratos Idreos, Manolis Koubarakis, Wolfgang Nejdl. In S. Staab and H. Stuckenschmidt (eds.): Semantic Web and Peer-to-Peer, Springer Verlag, 2004 [64].
- *Publish/Subscribe for RDF-Based P2P Networks.* By Paul - Alexandru Chirita, Stratos Idreos, Manolis Koubarakis, Wolfgang Nejdl. In Proceedings of the 1st European Semantic Web Symposium (ESWS), Heraklion, Greece, 2004 [65].
- *Personalized Reputation Management in P2P Networks.* By Paul - Alexandru Chirita, Wolfgang Nejdl, Mario Schlosser, Oana Scurtu. In Proceedings of the Trust, Security and Reputation Workshop held at the 3rd International Semantic Web Conference, Hiroshima, Japan, 2004 [68].
- *Knowing Where to Search: Personalized Search Strategies for Peers in P2P Networks.* By Paul - Alexandru Chirita, Wolfgang Nejdl, Oana Scurtu. In Proceedings of the 1st P2P Information Retrieval Workshop held at the 27th ACM International SIGIR Conference on Research and Development in Information Retrieval, Sheffield, United Kingdom, 2004 [69].

Acknowledgements

First, I would like to thank Prof. Dr. Wolfgang Nejdl. Not because every Ph.D. student starts with acknowledging his supervisor's work, but because in my case, his attentive guidance and continuous counseling during my master's thesis were decisive in even making me consider pursuing a Ph.D. It was an excellent choice. Wolfgang taught me how to organize my thoughts, how to approach and solve a problem, how to market my results. Born in the same day as my father, he went beyond such usual support: He helped both me and my wife to have a growing career, to accelerate our immigration process, to enjoy our lives.

I would also like to thank Prof. Dr. Ricardo-Baeza Yates for his kindness, for his useful research commments, and for hosting me at Yahoo! Research in 2006.

I am grateful to Prof. Dr. Klaus Jobmann and Prof. Dr. Heribert Vollmer for agreeing to be part of my dissertation committee.

A few other professors shaped my way through this point. Prof. Dr. Valentin Cristea believed in me and supported me continuously, both before and after my arrival in Hannover. Prof. Dr. Dan Iordache and Prof. Dr. Octavian Stanasila tutored my first research projects and recommended me for the great scholarship at Ecole Polytechnique in Paris.

In the beginning of my Ph.D. I had the luck of collaborating with Prof. Dr. Manolis Koubarakis, a great researcher, whose style had a visible influence upon my writing. Prof. Dr. Edleno de Moura, Prof. Dr. Pavel Calado, and Andre Carvalho, my hospitable hosts at the Federal University of Amazonas in Brazil, provided me with an extremely helpful experience with real-life search engines. My good friend Carlos Castillo paved my way within the Information Retrieval community, introducing me to the best researchers and believing in my skills.

I had the pleasure of working within a very high quality group, many of my colleagues and friends assisting me with indispensable research comments and

suggestions, as well as with a nice and friedly atmosphere. In particular, I would like to thank to all those younger Ph.D. and master students who reckoned my knowledge and chose to work with me. It would have been much harder without them.

Last, but definitely not least, I am forever grateful to my family. To my wife, for standing by me along this Ph.D., for enduring the distance when one of us had to be away and for supporting me in my initiatives. To my mother, for her excellent guidance through my development and education, but also for her extreme care. To my grandfather, who would have liked to see this, for shaping my intellectual skills since early childhood. To my father, for being there whenever I needed him. To my aunt Doina and uncle Bebe, my second pair of parents, for always caring for me. To Angi, Sica, and grandpa', for their great love.

Contents

Chapter 1

Introduction

The amount of information available in everyday life has grown exponentially during the previous century. From a very difficult access to knowledge in the early 1900, mankind reached a high state of development within just one hundred years, being now able to locate tons of data from all over the planet.

It all started around 1830, when Carl Friedrich Gauss electrified for the first time binary information in his telegraphy experiments. Though this moment is considered the birth of digital media (note that manual binary counting has been in practice at least since 2000 BC from the Babylonian Empire), it took until the Second World War to exploit this extremely valuable discovery at its entire capabilities, when computers began to be utilized on a somewhat larger scale of applications. In the same time, the amount of available storage solutions has grown as well. In 1928 Fritz Pfleumer invented the magnetic tape, a visible progress towards storing digital media, which was however fastly replaced around 1973 by floppy disks (invented in 1969 by David Noble from IBM), which at their turn were replaced by the much larger CDs (invented in 1985 by Kees Immink and Toshitada Doi from Philips and Sony) and DVDs (introduced in 1996-1998 by a large consortium of top media companies). Thus, more and more data has become digitalized. Almost all reports, presentations, media such as images or movies, books, etc. are now also produced in a digital format, making information distribution a much easier task. The already existing non-digital media such as printed books or articles is now slowly being digitalized as well, in order to provide a much faster access to it. Finally, along with the invention of the World Wide Web (1990-1991), it became trivial to even send such data almost anywhere on Earth within seconds. In fact, nowadays, for some people, if a document does not

exist in the Web, then it does not exist at all.

Clearly this strong scientific progress has brought huge benefits, which were probably fiction just one hundred years ago. Yet which challenges does it bring to us? There are plenty, but two of them have proved themselves to be really important: First, as so much information is now being available in digital format, when creating a data collection on some specific topic one needs to *collect* a large enough set of documents so as to cover most if not all the interests (or sub-topics) available for that subject. In some local environments such as enterprises, this task may not seem difficult. However, as soon as we think of larger corpora, such as topic-oriented news, or especially the entire World Wide Web, then gathering all documents therein becomes suddenly impossible. Second, once this usually overwhelming amount of data has been accumulated, one would need to locate those documents which are helpful in solving some given task. This is how the need for *search* appeared. Searching by itself deals with identifying the relevant documents within the corpora, given a specific user query. Yet this is still not sufficient. In large collections such as the Web, there will be millions of such documents. Therefore, an additional technique is necessary: *Ranking*. Ranking is the process of positioning items (e.g., documents, individuals, groups, businesses, etc.) on an ordinal scale in relation to others. This way one does not need to browse through the entire relevant output, but rather only look at the already identified best items.

The software that gathers and searches digital data is known under the broad name of *Search Engine* and the science seeking to design better algorithms for search engines is called *Information Retrieval*. IR is a broad interdisciplinary field, drawing on many other disciplines. It stands at the junction of many established research areas and draws upon cognitive psychology, information architecture, information design, human information behavior, linguistics, semiotics, information science, computer science, librarianship and statistics. The invention of the World Wide Web and the subsequent development of Web Search Engines has made IR an extremely popular research field, especially since a lot of new and highly interesting problems appeared together with this vast amount of data: How to crawl a large amount of Web pages in an efficient way? How to rank search results? How to personalize the search experience? How to suggest better queries to assist the user in search? How to cluster search results? And so on.

This thesis is about **Ranking in Large Scale Information Systems**. Research for efficient ranking algorithms is necessary for quite a lot of such application environments. Many examples can be given: The World Wide Web, Enterprise Networks, Digital Libraries, Social Networks, Peer-To-Peer Networks, Personal Information Repositories, Email Inboxes, etc. For all these, current ranking algo-

rithms are still rather poor or even inexistent, although in the same time they are more and more necessary, due to the extreme increase in the amount data stored and searched for each particular scenario.

All the algorithms we propose focus on **context oriented ranking**. For the case of excessively large media, such as the World Wide Web, Enterprise or Peer-To-Peer networks, this comes in the form of *personalization*. In these environments there is so much data that no matter which query is issued to the search engine, no matter which generic information organization algorithm is developed, thousands if not millions of matching items will be found[1]. As ranking alone is not sufficient to solve this problem, and as nobody has time to look into this many relevant results, an additional dimension is introduced, namely the specific preferences of each subject utilizing the system. For the case of emerging large media, such as PC Desktops or even Email Inboxes, though different, the situation is becoming more and more similar. First, Desktop search would commonly return several hundreds of results nowadays, which is again too much to browse through. In the same time many people receive dozens of emails per day, many of which are spam. Thus, even in this relatively small environment, some sort of item ranking measure would be useful in order to prioritize which incoming emails to read sooner, later, or not to read at all. Second, as the content addressed by these media is highly personal and heterogeneous across users, personalization is implicitly included in each application, i.e., each subject receives a different ranking, relative to her own data.

Besides being no longer manageable with current search mechanisms, the above mentioned information systems also exhibit another interesting commonality: They are all *Social Systems*. This might be obvious for macro-systems involving multiple users, as for example the Web, or social networks developed on top of email exchanges within a community. Yet the very same characteristics can be identified when performing this analysis at the micro-system level of single users! The glue around all these two perspectives is the *Power Law* [196], also known as "the rich get richer" law, which says that very few resources (i.e., persons, Web pages, etc.) are highly important across a collection, while all others are almost not important at all. For example, at the macroscopic level, the distribution of in-degree of Web pages follows a power law [34], just as the distribution of human social acquaintances [225]. As argued above, the same occurs at the microscopic level of single users. The distribution of time spent reading personal files on the Desktop follows again a power law [66], and so does the frequency of English words one would use in her personal documents [222].

In this thesis we provide a twofold contribution to the Information Retrieval field.

[1]Exceptions can be found, of course.

First, we identify those application areas in which a context oriented ranking is missing, though extremely necessary in order to facilitate a qualitative access to relevant resources. Second, for each of these areas we propose appropriate ranking algorithms which exploit their underlying social characteristics, either at the macroscopic, or the microscopic level. We achieve this by utilizing **link analysis** techniques, which build on top of the graph based representation of links between resources in order to rank them, or simply to identify social patterns relative to the investigated data set.

The thesis is organized around the applications of link analysis for ranking which we tackled, as follows: **Chapter 2** introduces us into the realm of ranking. We concentrate mostly on link analysis ranking algorithms, and especially on a description of PageRank [172] and HITS [143]. Once these have been presented, we also briefly describe the various link analysis ranking algorithms that followed them, as well as some of the other evidences used by search engines when ranking Web pages. In the end we give an overview of other information systems in need of ranking facilities, and motivate why they were left out from our investigation.

The three subsequent chapters describe each our specific contributions within the three emerging application areas we investigated. They all start with an introduction to the area, followed by a comparison with the related work specific to that particular domain. Subsequently, each proposed algorithm is first presented and explained, and then empirically evaluated. We conclude each chapter with a discussion on the pluses and minuses of each proposed algorithm, as well as on the possible further steps in the area.

In **Chapter 3**, we start from the motivation that current Desktop ranking algorithms are using only pure textual information retrieval techniques, which are more than 25 years old and which can no longer cope with the current information flow. We thus argue that Desktop search output should be ranked using a proper, specialized algorithm, based on specific indicators for this environment. As a solution, we propose two new approaches, both built on top of Desktop usage analysis: First, Section 3.3 investigates a technique in which only some specific Desktop activity contexts are studied and considered to confer ranking value to the resources associated to them. Then, we generalize this approach in Section 3.4 by considering all user accesses to Destktop items as relevant for ranking.

Chapter 4 dives into another central application of digital information management, Spam Detection. We tackle two environments which both suffer from the spam problem extensively: Email and Web. The initial half of the chapter (Section 4.3) presents our social network based reputation scheme for email addresses, together with a set of experiments proving its efficiency in combating the spam issue. These algorithms make a smooth transition from the local in-

formation environment (i.e., personal files, including emails, as stored on the PC Desktop) towards global milieus such as the Internet (represented here by the social networks constructed via the exchanges of emails). The second half of the chapter (Section 4.4) moves us even further towards global data management and introduces our site level approach to Web hyperlink spam detection.

Chapter 5 proposes several new approaches to better create and exploit user profiles for personalized Web search. We start with a discussion on how large scale taxonomies could be employed for both these goals, i.e., tailoring the Web search output according to user's interests and analyzing previous user actions in order to define a good profile. Though this technique yields very good results, it still requires sharing a small amount of personal information with the search engine. Therefore, in the second part of the chapter we concentrate on generating fully secure user profiles for the same task of Web search personalization. We extract preferences from user's Personal Information Repository and design an algorithm which achieves very good performance without sharing any private information with the search provider.

Chapter 6 concludes the thesis with an enumeration of the contributions we brought to the Information Retrieval research community, while also discussing possible future research directions and open challenges associated to these topics.

Chapter 2

General Background

Ranking has been widely investigated in search engine literature, both for the task of Information Retrieval per se, as well as for other closely related tasks. This chapter will therefore be split in two parts: First, we will present the inner details of search engine ranking, putting the focus on link analysis methods, as they are the foremost important approach for Web IR, and we argue, for quite several other environments as well. This part is especially important, as it will introduce the methods which we will build upon throughout the thesis. Once these have been introduced, in the second part of the chapter we will move towards discussing some of the already existing applications of ranking for other purposes than Web search, and we will motivate their exclusion from our study.

2.1 Ranking in the Web

2.1.1 Brief Introduction to Search Engines

Typical search engines consist of two major modules: A crawler and a searcher. Crawlers are assigned with the pre-processing task of gathering the search data into a local index (see Figure 2.1). They perform the following operations:

- *URL Listing*: Maintain a list of already visited URLs, as well as of the URLs which are to be visited in the future.
- *URL Retrieving*: Fetch from the Web new URLs received as input from the Listing Module.

- *URL Processing*: Process each visited URL in order to extract (1) its outgoing hyperlinks, which are then transferred to the URL Listing Module for further processing (i.e., filtering the already visited ones and planning the new ones for future retrieval), and (2) its indexable content, usually textual data, which is sent towards the Format & Store Module.
- *Data Format and Store*: Arrange the indexable data into a special format, compress it, and store it into the local index.

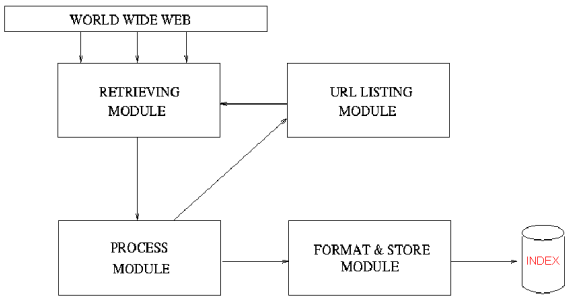

Figure 2.1: Crawler Architecture.

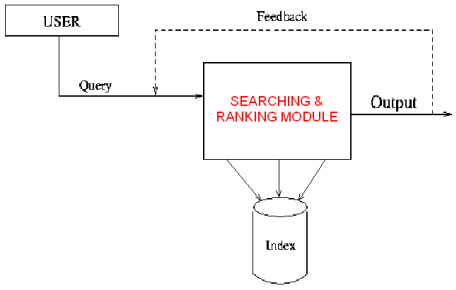

Figure 2.2: Searcher Architecture.

Once a reasonably large amount of Web pages has been collected, the user can start searching it. This is performed through a simple interface in which query keywords are entered (see Figure 2.2). Afterwards, the *Searching and Ranking Module* first inspects the index to gather the hits, i.e., the pages containing the user selected keywords, and then orders these hits according to different ranking criteria and displays them to the user as such.

The remainder of this section will mostly present a detailed look into link analysis ranking, and especially into PageRank and HITS, as they represent the foundation algorithms of this technique on the World Wide Web. In the end, we will briefly introduce some other measures used by search engines to order their output.

2.1.2 Link Analysis Ranking

Link analysis has been first utilized in the 1950's in a slightly different scenario: Citation Analysis (see for example the work of Katz [137] or Garfield [103, 104]). Just like in the Web, the need for citation analysis came because of the fastly growing amount of information, in this case coming from scientific conferences and journals. Thus, a publication venue evaluation technique was necessary in order to ease the identification of qualitative articles [105]. A set of interesting features were discovered, as for example the fact that few journals receive most

citations, while many journals receive few or even no citations at all [106]. When plotted, this turned into something which was yet to become a quite famous distribution: A power-law.

Power law models have been analyzed and rigorously defined within the same period, the first representative effort in this direction being that of Herbert Simon [196]. When defined over positive integers (the most common case), power-law distributions are characterized by having the probability of value i proportional to $1/i^k$, with k being a small positive real number. Nowadays, not only in the Web, but also in much wider contexts like social interactions or usage of natural language, the power law seems to dominate most of the discovered phenomena. Examples are plenty, from the distribution of in- and out-degree of Web pages [34, 149, 6, 19], to that of the frequency of words in English [160, 222], of social acquaintances [225], or even of oligonucleotide sequences within DNA [166]. Most important for us is the fact that the identification of power laws was very beneficial for Web searching and ranking [33, 43, 47, 150, 200]. We will thus now proceed to reviewing the most popular link analysis ranking algorithms, many of which yield highly qualitative results especially due to the power-law nature of their input.

PageRank

Preliminaries. Link analysis algorithms are founded on the representation of the Web as a graph. Hereafter we will refer to this graph as $G = (V, E)$, where V is the set of all Web pages and E is the set of directed edges $< p, q >$. E contains an edge $< p, q >$ iff a page p links to page q. $I(p)$ represents the set of pages pointing to p (in-neighbors) and $O(p)$ the set of pages pointed to by p (out-neighbors). We denote the *p-th* component of $\mathbf{v}$ as $v(p)$. Also, we will typeset vectors in bold and scalars (e.g., $v(p)$) in normal font. Finally, let A be the normalized adjacency matrix corresponding to G with, $A_{ij} = \frac{1}{|O(j)|}$ if page j links to page i and $A_{ij} = 0$ otherwise.

Description. PageRank [172, 33] computes Web page scores by exploiting the graph inferred from the link structure of the Web. Its underlying motivation is that pages with many backlinks are more important than pages with only a few backlinks. As this simple definition would allow a malicious user to easily increase the "importance" of her page simply by creating lots of pages pointing to it, the algorithm uses the following recursive description: "A page has high rank if the sum of the ranks of its backlinks is high". Stated another way, the vector **PR** of page ranks is the eigenvector corresponding to the dominant eigenvalue of A.

Given a Web page p, the PageRank formula is:

$$PR(p) = c \cdot \sum_{q \in I(p)} \frac{PR(q)}{\|O(q)\|} + (1-c) \cdot E(p) = c \cdot \sum_{q \in I(p)} \frac{PR(q)}{\|O(q)\|} + \frac{(1-c)}{\|V\|} \quad (2.1)$$

Theoretically, A describes the transition probabilities associated to a Markov chain. It is known that a finite homogeneous Markov chain which is irreducible[1] and aperiodic[2] has a unique stationary probability distribution π, depicting the end probabilities for the Markov chain to reach each specific state. The chain converges to this distribution no matter which is the initial probability distribution π_0. However, some Web content such as PDF articles have no out-links, and thus their corresponding column in A would consist only of 0 entries, making the Markov chain associated to the Web graph a reducible one. These pages are actually absorbing states of the Markov chain, eventually sucking all the PageRank into them. It is also trivial to show that the above mentioned Markov chain is a periodic one. These are both reasons for introducing a dumping factor $c < 1$ (usually set to 0.85) [32], which in fact also nicely models an intuitive description of PageRank: A random Web surfer will follow an outgoing link from the current page with probability c and will get bored and select a different page with probability $(1-c)$. Note that this model does not include the "BACK" button [155], but even so, it was proved in practice to yield very good results. Finally, one could use other non-uniform distributions for $\mathbf{E}$ if biasing or personalization is desired onto a given set of target pages (see also Equation 2.1, whose right hand side expression is derived assuming an uniform bias on all input pages).

Convergence and Stability Properties. The properties of **PR** are easily explained by the eigenvalues of A. Let us start from the well-known Perron-Frobenius theorem, as well as a proposition:

Theorem 1 *For any strictly positive matrix $A > 0$ there exist $\lambda_0 > 0$ and $x_0 > 0$ such that:*

1. $A \cdot x_0 = \lambda_0 \cdot x_0$;
2. *if $\lambda \neq \lambda_0$ is any other eigenvalue of A, then $|\lambda < \lambda_0|$;*
3. *λ_0 has geometric and algebraic multiplicity 1.*

Proposition 1 *Let $A > 0$ be a strictly positive matrix with row and column sums $r_i = \sum_j a_{ij}$, and $c_j = \sum_i a_{ij}$. Then, the "Perron-Frobenius" eigenvalue λ_0 is*

[1] A Markov chain is *irreducible* if any state can be reached from any other state with positive probability.

[2] A Markov chain is *aperiodic* if for any state i the greatest common divisor of its possible recurrence times is 1.

limited by:

$$\min_i r_i \leq \lambda_0 \leq \max_i r_i, \text{ and } \min_j c_j \leq \lambda_0 \leq \max_j c_j. \tag{2.2}$$

It is straightforward to see that for any stochastic matrix, all r_i are 1, and thus $\lambda_0 = 1$. However, an even more interesting eigenvalue is the second highest one. It is known that the asymptotic rate of convergence of the power method (i.e., Equation 2.1) is governed by the degree of separation between the dominant and the closest subdominant eigenvalues [155]. For our case, it has been shown that the closest subdominant eigenvalue is exactly c [119, 153]. Consequently, if we used $c = 0.85$ and aimed for an accuracy of 10^{-4}, we would need to run the computation process for 43 iterations, since $\lambda_2^{43} = 0.85^{43} < 10^{-4}$.

Another interesting property of PageRank is its *score stability* [157] on the class of all directed graphs. Intuitively, if a small modification (e.g., one link is deleted or added to some page p) occurs in the hyperlink structure of its underlying graph, then the difference between the new score $\tilde{PR}(p)$ and the old one $PR(p)$ has an upper bound close to 0. PageRank is also monotonic, i.e., adding a link towards a page can only increase its score, and therefore its rank [54]. However, Lempel and Moran [159] showed that all these properties do *not* imply *rank stability*: Even though a small change in the input Web graph results in a small score difference, it might be the case that the corresponding page is strongly promoted or demoted across the overall rankings.

Finally, we note that the distribution of the resulting PageRank values follows a power-law only for some particular values of the damping factor [21], and thus a careful selection of c is usually very important for the success of the application exploiting the ranking algorithm.

Dangling Nodes. As we have seen, pages with no out-links (also known as "dangling nodes") are quite problematic for PageRank. Even with the dumping factor in place, they may still have a negative influence upon the rankings. Thus, several solutions have been proposed. In the original paper [172], the authors suggested to remove them and calculate the PageRank only on the Web graph without dangling pages. Similarly, Kamvar et al. [133] proposed that after having calculated PageRank without the dangling nodes, to add them back in for several additional iterations. This approach was also suggested in [32, 120], where the authors remarked that this seems preferable to keeping them in the calculation. Note that the process of removing dangling nodes may itself produce new dangling nodes, and should therefore be repeated iteratively until no dangling nodes remain. It is interesting to note that this removal procedure seems to terminate rather quickly when applied to the Web. Finally, Eiron et al. [92] investigated the

possibility to jump to a randomly selected page with probability 1 from every dangling node, approach which seems to be the best choice so far.

Implementation Optimizations. There is a large amount of work on optimizing the computation of PageRank, mainly due to the extremely large size of its input data, the Web graph. Arasu et al. [12] suggested for example using the Gauss-Seidel method instead of the power iteration. Kamvar et al. [134] applied Aitken's Δ^2 quadratic extrapolation method obtaining an increase in speed of up to 300%. The same group also found small speed improvements by evaluating convergence over individual elements of PageRank [193] in order to proceed with the computation only for the non-converged nodes. Chien et al. [54] investigated the possibilities to update PageRank given some small changes in the Web graph. Chen et al. [51] proposed several I/O effective implementation approaches for the power iteration over large graphs. Finally, Langville and Meyer [154] proposed to move the dangling nodes towards the bottom of the PageRank matrix and showed this technique to further improve its computation time. Obviously most of these works focus on time optimizations. However, we note that other aspects have been tackled as well, such as improving the space complexity of the algorithm [117], or calculating its output under missing data [3], or enhancing the ranking procedure to allow for both horizontal (i.e., topology based) and vertical (i.e., topical based) processing [82].

Another strongly investigated research area is the parallelization of PageRank. Existing approaches to PageRank parallelization can be divided into two classes: Exact Computations and Approximations. For the former ones, the Web graph is initially partitioned into blocks: grouped randomly (e.g., P2P PageRank [189]), lexicographically sorted by page (e.g., Open System PageRank [195]), or balanced according to the number of links (e.g., PETSc PageRank [111]). Then, standard iterative methods such as Jacobi or Krylov subspace [111] are performed over these pieces in parallel, until convergence. The partitions must periodically exchange information: Depending on the strategy this can expose suboptimal convergence speed because of the Jacobi method and result in heavy inter-partition I/O. In fact, as the Jacobi method performs rather slow in parallel, we modified the Gauss-Seidel algorithm to work in a distributed environment and found the best speed improvements so far (see Kohlschütter, Chirita and Nejdl [145]).

When approximating PageRank, the idea is that it might be sufficient to get a rank vector which is comparable, but not equal to PageRank. Instead of ranking pages, higher-level formations are used, such as the inter-linkage between hosts, domains, server network addresses or directories, which is orders of magnitudes faster. The inner structure of these formations (i.e., the page level ranking) can then be computed in an independently parallel manner ("off-line"), by combining

the local rank of each page with the global rank of the higher level entity it belongs to, as in BlockRank [133], SiteRank [218, 1], the U-Model [36], ServerRank [213] or HostRank / DirRank [92].

Derived Algorithms. There are a lot of publications proposing either alternatives to PageRank, or small modifications to it. Some of the latter ones have already been mentioned above (e.g., [92]), as they are intended to achieve better computation performance. We will thus focus here only on those papers exhibiting bigger differences when compared to PageRank, either in terms of methodology or of the end purpose of the computation. Baeza-Yates and Davis [16] improve PageRank quality by giving different weights to links as a function of the tag in which they were inserted, of the length of the anchor text, and of the relative position of the link in the page. Feng et al. [96] calculate a ranking exclusively over the Web sites, rather than the more granular pages. They start in a similar manner to BlockRank [133], computing PageRank within each site separately, but then this information is utilized to derive a stochastic coupling between Web sites, which is later applied in a regular power iteration at the Web site level. Moreover, other optimized Web ranking algorithms include the work of Upstill et al. [207], who argue that ranking by the in-degree of pages is usually enough to approximate their quality, as well as the work of Abiteboul et al. [2], who attempt to compute page reputations (PageRank approximations) directly at crawling time. In this latter paper, the iterations of the power method are achieved through the re-discovery of new links towards already visited pages, as well as through crawl updates. Several crawling strategies are identified, but the greedy approach seems to be closest to the power-law model of the Web. The interesting aspect of this work is that page scores are updated as the Web is crawled over and over again, thus implicitly coping with its volatility.

Another PageRank related research direction it to bias its scores towards the topic associated to each user query. The most popular work is that of Haveliwala [118], who builds a topic-oriented PageRank, starting by computing off-line a set of 16 PageRank vectors biased[3] on each of the 16 main topics of the Open Directory Project[4] (ODP). Then, the similarity between a user query and each of these topics is computed, and the 16 vectors are combined using appropriate weights. Rafiei and Mendelzon [179] include the topics covered by Web pages into the reputation algorithm, each topic being represented by a set of terms. Given a topic, the pages covering it (i.e., containing its descriptive words) are identified and used in the ranks calculation. The approach is not feasible, due to the practically infinite amount of existing topics. Nie et al. [170] had the better idea of distributing

[3]Biasing is obtained by setting higher values for the targeted pages within the **E** vector.
[4]http://dmoz.org

the PageRank of a page across the 16 ODP topics it contains. This way, the importance flows according to the text content of the source and target pages, and all 16 topic oriented rankings are generated at once[5].

Finally, there exist also some works moving a bit beyond PageRank. Baeza et al. [14] for example investigated various new damping functions for link analysis ranking, showing them to result in rather similar quality to PageRank, while being computationally much faster. Tomlin [205] proposed to use the richer network flow model instead of the now common Markov chain approach to computing PageRank. Last, but not least, we proposed HubRank (see Chirita, Olmedilla and Nejdl [72, 73]), which biases PageRank onto hubs, thus nicely combining the authoritative computation of PageRank with a hub measure for Web pages. Moreover, we showed this approach to yield better qualitative results than regular PageRank over a set of toy experiments.

HITS

Description. Kleinberg's HITS [143] was the first efficient link analysis ranking algorithm for the Web. It builds upon the idea of computing two scores for each page in a Web community, namely a hub score and an authority score. Generally, a hub is a page pointing to many other authoritative pages, whereas at the opposite end, authorities are pages containing valuable information pointed to by many hubs. The algorithm starts from a set $\mathbf{R}$ of pages with high PageRank, as returned by a search engine. This set is first extended into $\mathbf{R_{ext}}$ with all the pages pointing to, as well as pointed by pages from $\mathbf{R}$. Then, given that each of these pages has been assigned an initial authority score a_i^0 and a hub score h_i^0, HITS refines their scores using the following iteration procedure:

$$a_i^k = \sum_{<p,q>\in E} h_p^{k-1}; \qquad h_i^k = \sum_{<p,q>\in E} a_q^{k-1} \tag{2.3}$$

For convergence purposes, after each iteration the values within $\mathbf{a}$ and $\mathbf{h}$ need to be normalized to sum to 1 (as otherwise they would continuously increase). Written in matrix form, if L is the adjacency matrix of the graph underlying the pages from $\mathbf{R_{ext}}$, the same equations become:

$$\mathbf{a}^k = L \cdot L^T \cdot \mathbf{a}^{k-1}; \qquad \mathbf{h}^k = L^T \cdot L \cdot \mathbf{h}^{k-1} \tag{2.4}$$

[5]Of course, this approach limits the application to computing topical oriented rankings for only a small number of topics.

Just as with PageRank, it is straightforward to notice that $\mathbf{a}$ converges to the dominant eigenvector of $L \cdot L^T$, and $\mathbf{h}$ converges to the dominant eigenvector of $L^T \cdot L$. In fact, Ding et al. [83] have discovered another interesting property of these vectors: There is a direct relationship between HITS' authority matrix $L \cdot L^T$ and the co-citation matrices used in bibliometrics; similarly, the hub matrix $L^T \cdot L$ is related to co-reference matrices.

Since HITS is usually constructed around the Top-K (usually 200) pages returned as output to some user query, it is usually computed over less than 20,000 pages, and is thus very fast. Moreover, one needs to compute only one of the above mentioned eigenvectors: For example, given the authority vector (i.e., the eigenvector of $L \cdot L^T$), then the corresponding hub vector is given by $\mathbf{h} = L \cdot \mathbf{a}$.

A series of limitations made HITS less successful than PageRank. First of all, its results are topic drifted, i.e., they are focused around the main topic of the input graph. This problem was solved by Bharat and Henzinger [27] through a weighting of the authority and hub scores according to the relevance of each page to the initial user query. Gibson et al. [110] exploited this problem in order to infer the topics residing within different subsets of the Web graph. A second problem is HITS' susceptibility to spamming. It is fairly easy to construct a very good hub, and subsequently to push the score of a target authority page. Also, HITS is neither rank stable, nor score stable [157, 159].

Derived Algorithms. HITS was also studied extensively and many improved variants of it have been proposed. However, since our work is only partially related to HITS, we will review here just some of its relevant follow-up algorithms. Randomized HITS [169] is a two-level reputation ranking approach, combining the random surfer model from PageRank with the concepts of hubs and authorities from HITS in order to achieve a rank stable algorithm for calculating hub and authority scores over a given graph. Similarly, SALSA [158] adopts two Markov chains for traversing the Web graph, one converging to the weighted in-degree of each page, for authority scores, and the other converging to its weighted out-degree, for hub scores. The algorithm is thus no longer dependent on Tightly Knit Communities of pages (as HITS is), but is still vulnerable to many forms of spam. Other variants have been proposed by Tsaparas [206] and Borodin et al. [29], though their work is more important due to the theoretical analysis therein, rather than the qualitative improvements of the algorithms proposed. Finally, we also note that a lot of research on the HITS algorithm has been performed within the IBM CLEVER project [171].

Other Link Analysis Ranking Algorithms and Beyond

Though there are many other approaches to ranking pages in the World Wide Web, we would like to briefly discuss here only the most important one of them: Machine Learning. As Web ranking becomes more and more complicated, with increasingly more input features being necessary, Machine Learning seems to gain momentum.

The most important approach in this category is RANKNET [39, 182], apparently the ranking mechanism used by the Microsoft Live search engine. Its underlying idea is to use machine learning to combine a large amount of features of Web pages, starting from the already common link based ones, and up to visiting statistics for different pages, as well as textual evidences. The authors show this technique to yield better ranking results than PageRank.

Another technique which has been investigated for the Web ranking purpose is Latent Semantic Analysis [80]. Cohn and Chang [75] use it to propose a probabilistic model to estimate the authority of documents in the Web. Unlike the eigenvector based solutions, they apply Hoffman's Probabilistic LSI model [123] over a Document x Citation matrix and obtain better document quality estimates. However, their performance is dependent on the chosen starting set and may get stuck in local optima with poor overall results. Also, it is not clear how fast this algorithm would compute on Web size input data.

2.1.3 Other Features used for Web Ranking

This thesis builds upon the link analysis methods presented in the previous section. Nevertheless, we have seen that such information is not sufficient. First and foremost, without text analysis, it would be nearly impossible to accurately identify the documents best matching a user query. The most employed technique for this matter is the Vector Space Model [188], according to which both queries and Web pages are represented as bags of words, weighted by their Term Frequency multiplied by Inverse Document Frequency. Many extensions are possible in the Web environment, very important being the differentiation of terms based on the mark-up used around them (i.e., title, bold, etc.), and the inclusion of anchor text in the actual Web page.

A second source of ranking data is represented by the query logs. In fact, in the pre- link analysis era, Yahoo! used to rank its output utilizing the number of clicks obtained by each URL when displayed as response to some input query. While its importance has now decreased, the click rate is still a highly relevant factor in Web search engines. Moreover, it can give a lot more information besides the actual

importance of documents. One could for example locate terms that frequently co-occur with some query, and thus automatically bias and improve the quality of the results list. Or, for ambiguous search requests, one could determine the "popularity" of each query interpretation and tailor the search output accordingly (e.g., the "Java" programming language is by far more popular than the coffee, or the island in the Pacific).

Third, we have the session specific features. These are usually based on mining the IP addresses of users. The most important one is the geographic location, as it is clear that different cultures imply different perspectives on a qualitative search result for most queries. Then, there is the time of day. Major global subject interests change over the day, ranging for example from news in the morning, to shopping in the evening. The same applies to the analysis of daily interests over the year. Finally, as people feel uncomfortable with sharing their search history, several simple personalization techniques have been developed, such as anonymously identifying each person using a cookie stored on her machine.

A lot of other sources of quality rating exist. It is believed that the major companies utilize over 500 such features. Some examples not covered by the above mentioned categories include the fact of being listed in hand crafted Web taxonomies such as the Open Directory, the age of each page, the amount and frequency of changes operated on it, etc.

2.2 Using Ranking in IR Applications

In this thesis we tackle the major emerging applications of link analysis for ranking. Nevertheless, there exist a few other utilizations of PageRank and alike. This section briefly surveys these additional interest areas, together with a discussion of their future success potential.

Social Network Ranking. As PageRank is strongly exploiting the social nature of humans, some authors proposed to develop social reputation metrics based on votes for (and sometimes also against) other individuals [124]. We have also pursued this goal for ranking people within Peer-To-Peer environments (see Chirita et al. [69], or Kamvar et al. [135]). However, the conclusions drawn from these analyses indicate that PageRank provides only a minor improvement for regular Peer-To-Peer tasks, such as known item search or generic keyword based search [57, 65, 68, 69].

Web Characterization. There has been quite a lot of research on Web characterization in the past, yet only few studies included a PageRank analysis as well. Panduragan et al. [174] were among the first to show that the importance of Web

pages (i.e., their PageRank) follows a power-law distribution. At the other end, Arasu [12] investigated several PageRank computational optimizations which exploit the Web structure discovered in previous characterizational studies. All in all, this field's importance has now decreased, as the wide span of Web analytical methods discovered so far seems to be sufficient for the current algorithmic necessities.

Ranking Concepts over the Semantic Web. Swoogle crawls the so-called Semantic Web seeking for any existing ontological instances. The located items are then ranked using a variant of PageRank [84] built on top of the links between the identified objects. The small size of the search engine and the current insuccess of the Semantic Web on a global scale make questionable the success of this otherwise interesting application.

Text Summarization and Classification. Based on the assumption that the macro social dynamics caught by PageRank could be in fact also present at the micro level of singular subjects, it was believed that the similarity between the sentences and / or documents authored by the same person also follows a power-law. More specifically, if we build links between our previously authored sentences / documents and weight them according to the level of textual similarity (in terms of overlapping words) between the connected nodes, then we obtain a graph shaped by a power-law degree distribution. Consequently, the most "representative" sentences can be used to summarize their underlying documents [93, 94, 81], or to group these documents into categories [100, 13]. This technique performs fairly well, yet still below the more powerful Natural Language Processing algorithms. We also applied such a micro social analysis onto another, more promising application area: Personal Information Management. More details are given in the next section.

Chapter 3

Ranking for Enhancing Desktop Search

3.1 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. Using this space, it is quite common to have over 100,000 indexable items within our Personal Information Repository (PIR). 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 searching and organization in personal repositories has received more and more attention during the past years. Thus, several projects have started to explore search and Personal Information Management (PIM) on the Desktop, including Stuff I've Seen [86], Haystack [178], or our Beagle^{++} [63].

Web search has become more efficient than PC search due to the boom of Web search engines and to powerful ranking algorithms like the PageRank algorithm introduced by Google[1]. The recent arrival of Desktop search applications, which index all data on a PC, promises to increase search efficiency on the Desktop (note that we use the terms Desktop and PIR interchangeably, referring to the personal collection of indexable files, emails, Web cache documents, messenger history, etc.). However, even with these tools, searching through our (relatively

[1] http://www.google.com

small set of) personal documents is currently inferior to searching the (rather vast set of) documents on the Web. This happens because these Desktop search applications cannot rely on PageRank like ranking mechanisms, and they also fall short of utilizing Desktop specific characteristics, especially context information. Indeed, Desktop search engines are now comparable to first generation Web search engines, which provided full-text indexing, but only relied on textual information retrieval algorithms to rank their results.

Most of the prior work in Personal Information Management has focused on developing complex, yet user friendly, systems for organizing and re-finding information using visualization paradigms, rather than ranking ones. This was in pursue of the hypothesis that all Desktop documents are equally important, and thus no ranking is necessary. In a more recent formulation, this was denoted "search = re-finding". In this thesis we advocate the contrary: We argue that some personal documents are actually much more important than others, and that users would search for these documents much more often than for any other ones.

We therefore have to enhance simple indexing and searching of data on our Desktop with more sophisticated ranking techniques. Otherwise, the user has no other choice, but to look at the entire result sets for her queries – usually a tedious task. The main problem with ranking on the Desktop comes from the lack of links between documents, the foundation of current ranking algorithms (in addition to TFxIDF metrics). A semantically enhanced Desktop offers the missing ingredients: By gathering semantic information from user activities, from the contexts the user works in[2], we build the necessary links between documents.

Within this chapter we propose to enhance and contextualize Desktop search by analyzing user's local resource organization structures, as well as her Desktop activity patterns. We investigate and evaluate in detail the possibilities to translate this information into a Desktop linkage structure, and we propose several algorithms that exploit these newly created links in order to efficiently rank Desktop items. Where applicable, we also utilize the same information in order to generate resource specific metadata, which is then employed to enhance Desktop search recall. We empirically show that all our algorithms lead to ranking results significantly better than TFxIDF when used in combination with it, thus making metadata and especially access based links a very valuable source of input to Desktop search ranking algorithms.

The chapter is organized as follows: We start in the next section with a review of the previous specific attempts to enhance Desktop search. In Section 3.3 we

[2]Studies have shown that people tend to associate things to certain contexts [129], and this information should be utilized during search. So far, however, neither has this information been collected, nor have there been attempts to use it.

describe and empirically evaluate our first ranking algorithm, which ranks personal items by analyzing exclusively several pre-defined user actions. We generalize this approach by considering all resource accesses within the algorithm from Section 3.4. In the end, we conclude the chapter with a discussion on the pluses and minuses of each technique, as well as on the possible next steps.

3.2 Specific Background

Though *ranking* plays an important role on the Web, there is almost no approach specifically aiming at *ranking* Desktop search results. Even though there exist quite a few systems organizing personal information sources and improving information access in these environments, few of the papers describing them concentrate on search algorithms. In this section we will first describe several such systems and discuss their approaches to Desktop search. Then, we will concentrate our attention towards some of the other existing personal information management systems, whose purpose was to provide means for organizing the local information, rather than searching it. Finally, we will briefly review the current industrial approaches for Information Retrieval at the PC Desktop level.

3.2.1 Ranking Algorithms for the PC Desktop

Very few works fall into this category. A very recent one, Connections [198], is probably the only system specifically targeted at enhancing Desktop search quality. Similar to us and to Haystack [4], they also attempt to 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, without a clear increase in output quality.

The ranking paradigm addressed to generic personal data collections has also been researched in the context of the Semantic Web. Aleman-Meza et al. [7] for example analyzed the importance of semantically capturing users' interests in order to develop a ranking technique for the large number of possible semantic associations between the entities of interest for a specific query. They defined an ontology for describing the user interest and used 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 interesting technique for ranking the results for a query on a semantic data set takes into consideration the

inferencing processes that led to each result [201]. In this approach, the relevance of the returned results for a query is computed based upon the specificity of the relations (links) used when extracting information from the knowledge base. The calculation of the relevance is however a problem-sensitive decision, and therefore task oriented strategies must be developed for this computation.

Finally, as current Information Retrieval literature falls short of providing valuable ranking mechanisms for the PC Desktop, most tools for this environment recur to traditional textual retrieval models, such as the Vector Space Model [17]. The only ordering criterion specific for personal information is to sort items by their recency, i.e., by the time difference between the current moment and their last access stamp. Clearly, this is a naïve technique, which gives valuable results only in particular cases.

3.2.2 General Systems for PIM

Several systems have been constructed in order to facilitate re-finding of various stored resources on the Desktop. Stuff I've Seen [86] for example 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, thumbnails and previews can be used to search for and present information, but no Desktop specific ranking scheme is investigated. Similarly, MyLifeBits [109] targets storing locally 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.

The Fenfire project [95] proposes a solution to interlink any kind of information on one's Desktop. That might be the birthday with the person's name and the articles she wrote, or any other kind of information. The idea is to make the translation from the current file structure to a structure that allows people to organize their data closer to the reality and to their needs, in which making comments and annotations would be possible for any file. Nevertheless, the purpose of this process has again no relation to *personal information retrieval*, i.e., searching and ranking on the Desktop.

Haystack [178] pursues similar goals as Fenfire. One important focus is on working with the information itself, not with the programs it is usually associated with. For example only *one* application should be enough to see both a document, and the email address of the person who wrote it. Therefore, a user could build her own links to Semantic Web objects (practically any data), which could then be viewed as thumbnails, Web pages, taxonomies, etc. The underlying idea of the

project was to emphasize the relationship between a particular individual and her corpus [4]. On the one hand, this is quite similar to our approach in the sense that it automatically creates connections between documents with similar content and it exploits activity analysis to extend the Desktop search results set. On the other hand, just like the previous articles, it does not investigate the possibilities to *rank* these results, once they have been obtained. Its follow-ups [127, 136] further explore the efficient organization of Desktop resources. They use an RDF database to store metadata about the various personal items, as well as about any connections between different Desktop data. Finally, Magnet [197] 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.

A smaller yet similar system, Semex [85], automatically generates associations between items on the Desktop in order to provide a meaningful context based local browsing. Interestingly, they also support on-the-fly integration of associations stemming from both personal and public data. Again, no ranking is discussed in their prototype whitepapers.

Lifestreams [102, 98] is an older Desktop organization system based on a time-ordered stream of documents meant to replace conventional files and directories. All its aspects, including query results, consist of substreams of the main Desktop usage stream, thus being rather different from the systems nowadays.

Hull and Hart [126] modified conventional PC peripherics (e.g., printers) to automatically store every processed document, thus providing search through any previously accessed document. They also use only traditional ranking techniques, such as ordering by date or TFxIDF. Though it does not describe the architecture of a system, the work of Ringel et al. [183] is also quite relevant for Desktop search applications: They suggested using timeline visualizations augmented with public and personal landmark events in order to display query results over an index of personal content.

Gnowsis [191, 192] and IRIS [53] create a "personal map" across various types of personal information objects. They allow users to annotate the files they accessed, as well as to manually establish links between them. This way, a semantic metadata repository is populated and provided as a basis to other, semantically enhanced Desktop applications, including search.

Finally, this work was developed in the context of the Beagle^{++} system [61, 63, 62]. There, we first produce a collection of metadata associated to each personal resource, as well as a linkage structure over the PIR, using similar methods as Gnowsis and IRIS. However, we also apply the results of this research for a specific task, namely developing Desktop searching and ranking algorithms. More details

about how links are collected and exploited in Beagle^{++} can be found in Sections 3.3 and 3.4.

3.2.3 Specific Applications aimed at Desktop Search Only

Desktop search applications are not new to the industry. Only the high interest in this area is new. For example, applications such as Enfish Personal[3] have been available since 1998, usually under a commercial license. As the amount of searchable Desktop data has reached very high values and will most probably also amplify in the future, the major search engines have recently given more focus to this area than the academia. Thus, several Desktop search distributions have been released for free (e.g., Google Desktop Search[4], MSN Desktop Search[5], etc.). Moreover, some providers have even integrated their Desktop search tool into the operating system, such as Apple[6]. The open source community has also manifested its interest in the area, the most prominent approaches being Gnome Beagle[7] (now also integrated into SuSE) and KDE KAT[8], developed within the Mandriva community. Many other commercial Desktop search applications exist (e.g., Copernic, Yahoo! Desktop Search, X1, Scopeware Vision, PC Data Finder, etc.), but as our main focus is to devise a Desktop ranking algorithm, rather than an entire search application, we will not dive into the particularities of each of these tools.

Most of the above mentioned applications target a very exhaustive list of indexed file types, including any metadata associated to them. They also update their index on the fly, thus inherently tracking any kind of user activity. However, all of them seem to only employ dates, or TFxIDF as measures to rank search results, without exploiting the usage information they have available. Therefore, they inherently miss the contextual information often resulting or inferable from explicit user actions or additional background knowledge.

3.3 Ranking by Tracking Specific User Actions

The vast majority of search activities at the PC Desktop level are navigational, in the sense that the user is trying to locate one or more items that were almost

[3] http://www.enfish.com/
[4] http://Desktop.google.com/
[5] http://toolbar.msn.com/
[6] http://www.apple.com/macosx/features/spotlight/
[7] http://www.gnome.org/projects/beagle/
[8] http://kat.mandriva.com/

surely stored somewhere in the personal information repository in the past. We argue however that the current abundance of Desktop data makes such a location impossible without any efficient ranking mechanism. On my Desktop for example, the query "Google ranking algorithm" (i.e., a three word query, strongly above the average query length of 1.7 terms for Desktop search [86]) would return about 500 results! More, this is definitely a clear query, with a very specific search goal in mind. This yields two motivations for bringing ranking into Desktop search: (1) One would surely not be willing to browse through all the 500 results until she finds the right document; (2) The fact of having so many outputs for such a clear query indicates that good query formulation is no longer sufficient in finding items within the personal information repository.

This chapter will propose a preliminary solution for Desktop ranking. We will first isolate several common user activity contexts and show how to exploit them for locating previously stored information. Then, we will put these usage areas together into a semantic ranking architecture meant to enhance the precision of Desktop search algorithms by personalizing on each user's regular file access patterns. The empirical results depicted in the last part of the chapter will prove our approach to yield visible improvements when seeking for items in the PIR.

3.3.1 Activity Contexts at the Desktop Level

In this section we overview the Desktop activity contexts we considered. We describe the metadata information that can be extracted from personal resources, as well as the implicit connections residing between them. The former enhance recall by including the relevant document in the results set even when the user query is poorly formulated, and the latter induce a PIR ranking which implicitly strongly enhances Desktop search precision.

Exploiting the Email Context

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

Context and Metadata. There are several aspects relevant to our email context. Sender and receiver fields of the email are clearly relevant pieces of information. Basic properties for this context are also the date when an email was sent, or the date it was accessed, the subject of the email and its body. The status of the email can be described as seen / unseen or read / unread. We also have a property of the type *reply_to*, which gives thread information and is useful to determine social network information in general, for example which people discussed which topic, etc. The *has_attachment* property describes a 1:n relation between the mail and its one or more attachments. The *to* and *from* properties connect to Class *MailAddress* which connects to Class *Person*. A *Person* is usually associated to more than one *MailAddress* instances. For attachments we also keep their connection to the email they were saved from, because when we search for an attachment we want to use all attributes originally connected to the email it was attached to (see the motivating scenario above). The *stored_as* attribute is the inverse relation of the *File:stored_from* property, which we describe later. Note that all these metadata should be generated automatically, while the user works, according to the schema depicted in Figure 3.1.

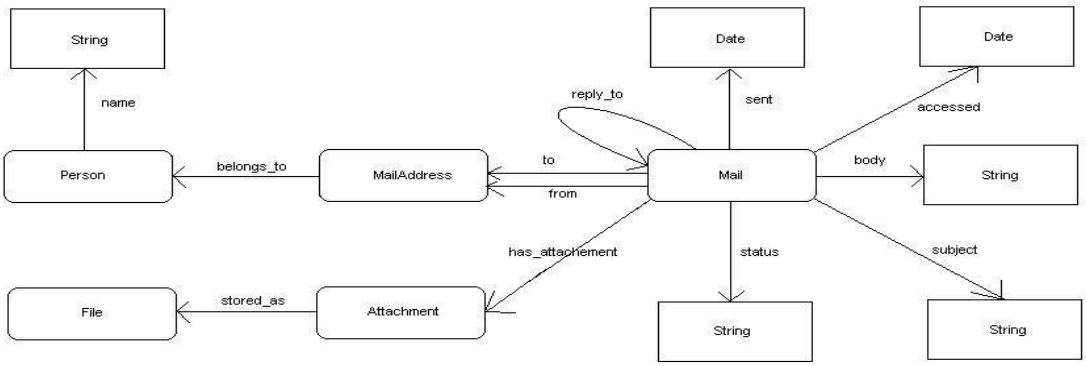

Figure 3.1: Email context.

Exploiting the File Hierarchy Context

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

Desktop search with a query like "pictures Germany" would assist in retrieving his Hannover pictures.

Context and Metadata. Obviously, our context metadata for files include the basic file properties like last date of access and creation, as well as the file owner. File types can be inferred automatically, and provide useful information as well (in our case, the file is of type "JPEG image data"). Additionally, a file might be a visited Web page which we stored on our computer or an attachment saved from an email. This *stored_from* property is of great importance because it represents information that current file systems miss, the provenance of information. We also keep track of the whole file path, including directory structure. Finally, we extend the strings used in the path name using WordNet [167], a lexical reference system which contains English nouns, verbs, adjectives and adverbs organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets. In our case, we use the following additional relationships to enrich the context of the stored file:

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

The complete file context ontology is also depicted in Figure 3.2.

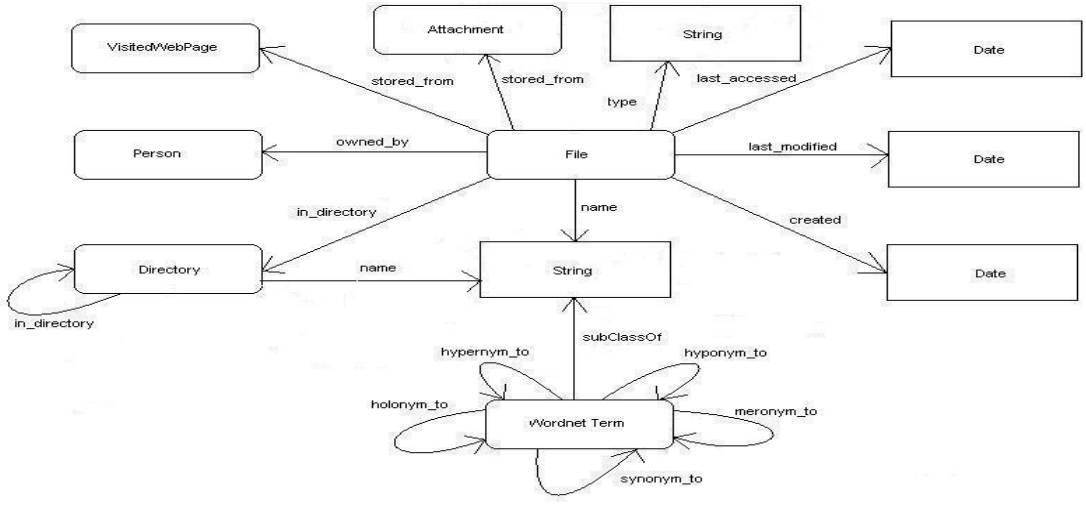

Figure 3.2: File context.

Exploiting the Web Cache Context

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

Context and Metadata. The central class in this scenario is *VisitedWebPage*. Upon visiting a Web page, the user is more interested in the links she has used on that page, rather than every possible link which can be followed from there. Thus, the metadata contains only the hyperlinks accessed for each stored Web page: (1) *departed_to* shows the hyperlinks the user clicked on the current Web page, and (2) *arrived_from* represents the page(s) the user came from. Also here, we have added properties related to the time of access and place of storage in the hard disk cache. For more specific scenarios, we can further define subclasses of this base class, which include scenario specific attributes, for example recording the browsing behavior in CiteSeer, which we will discuss in the next section.

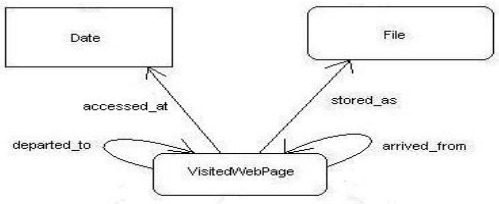

Figure 3.3: Web page context.

Exploiting the Browsed Publications Context

Scenario. This is an example of a more specific scenario. Suppose that Alice browses through CiteSeer for papers on a specific topic, following reference links to and from appropriate papers, and downloads the most important documents onto her computer. Now as soon as they are stored in one of her directories, her carefully selected documents are just another bunch of files without any relationships. They have completely lost all information present in CiteSeer, in this case which paper references or is referenced by other papers, and which papers Alice deemed important enough not only to look at but also to download. In our system we preserve this information and make it available as explicit metadata.

Context and Metadata. As discussed, stored files on today's computers do not tell us whether they were saved from a Web page or from an email, not to mention the URL of the Web page, out-going or in-going visited links and more specific information inferable using this basic knowledge and a model of the Web page context browsed, as discussed in our scenario. We therefore create a subclass of *VisitedWebPage* called *Publication*, and add suitable properties as described in Figure 3.4. The *Publication* class represents a cached CiteSeer Web page. It records the CiteSeer links traversed from that page using the *references* property, as well as the CiteSeer documents which the user visited before, using the *referenced_by* property. It is easy to notice that these pages represent a subset of the metadata captured by the *departed_to* and *arrived_from* relations. *PDF_file* and *PS_file* are subclasses of *File* and describe the format used to save the publication into the PIR. They are connected to *Publication* with subproperties of *stored_as*, namely *stored_as_pdf* and *stored_as_ps*.

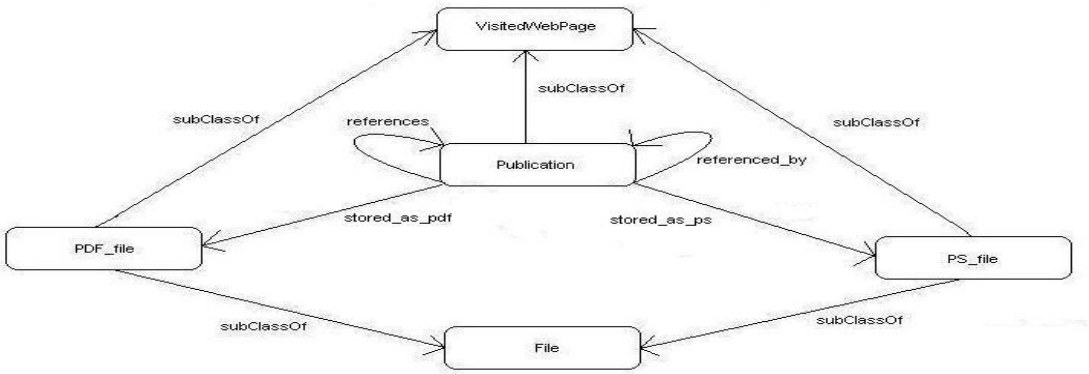

Figure 3.4: Browsed publications context.

Other Contexts Enhancing Desktop Search

Resources on the PC Desktop offer us a lot of additional sources of information. For example many multimedia files contain embedded metadata, such as the artist and title of a song saved as an mp3 file, or the date and time when a picture was taken. Some resources could be embedded into others, thus linking various Desktop contexts (e.g., a picture might be inserted in a text document describing a city). Similarly, our chat discussions with work partners from other locations might contain some valuable information as well, including related pictures or documents transmitted via the messaging application. All in all, the more contexts included in the Desktop search architecture, the broader will be its resource coverage and output quality.

3.3.2 A Context Oriented Architecture for Desktop Search

We will now present our 3-layer architecture for generating and exploiting the contextual metadata enhancing Desktop resources. At the bottom level, we have the physical resources currently available on the PC Desktop. Even though they can all eventually be reduced to files, it is important to differentiate between them based on content and usage context. Thus, we distinguish structured documents, emails, offline Web pages, general files[9] and file hierarchies. As discussed in the previous section, while all of them do provide a basis for Desktop search, they also miss a lot of contextual information, such as the author of an email, or the browsing path followed on a specific Web site. We generate and store this additional search input using metadata annotations, which are placed on the second conceptual layer of our architecture. Finally, the uppermost layer implements a ranking mechanism over all resources on the lower levels. An importance score is computed for each Desktop item, supporting an enhanced ordering of results within Desktop search applications. The complete architecture is depicted in Figure 3.5. In the next subsections we describe both its higher level layers following a bottom-up approach.

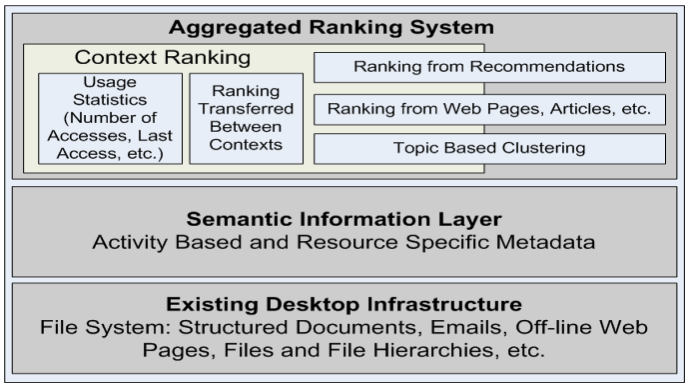

Figure 3.5: Contextual Desktop Ranking Architecture.

The Semantic Information Layer: Bringing the Contexts Together. People organize their lives according to preferences often based on their activities. Consequently, Desktop resources are also organized according to performed activities and personal profiles. Since most of the information related to these activities is lost within our current Desktop applications, the goal of the semantic information layer is to record and represent this data, as well as to store it as annotations associated to each resource. Figure 3.6 depicts an overview of the ontology that defines appropriate annotation metadata for the contexts we are focusing on.

[9]Text files or files whose textual content can be retrieved.

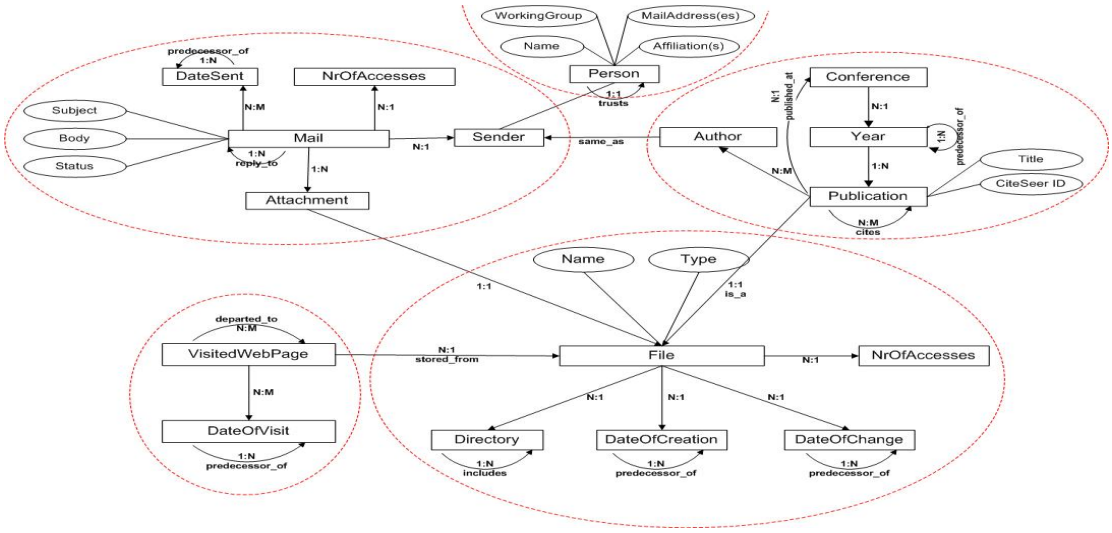

Figure 3.6: Contextual Ontology for the Desktop.

The main characteristic of our extended Desktop search architecture is metadata generation and indexing on-the-fly, triggered by modification events generated upon occurrence of file system changes. This relies on notification functionalities provided by the OS kernel. Events are generated whenever a new file is copied to hard disk or stored by the Web browser, when a file is deleted or modified, when a new email is read, etc. Depending on the type and context of the file / event, various appropriate software modules generate each specific type of metadata and export it into the Desktop search engine index.

The Contextual Ranking Layer. As the amount of Desktop items has been increasing significantly over the past years, Desktop search applications will return more and more hits to our queries. Contextual metadata, which provide additional information about each resource, result in even more search results. A measure of local resource importance is therefore necessary. In the following paragraphs we propose such a ranking mechanism which exploits the "popularity" of each item and the connections between user's activity contexts.

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

Then, we have to specify how these aspects influence importance. ObjectRank [18] has introduced the notion of authority transfer schema graphs, which ex-

tend 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. They extend our context ontologies with the information we need to compute ranks for all instances of the classes defined in the context ontologies.

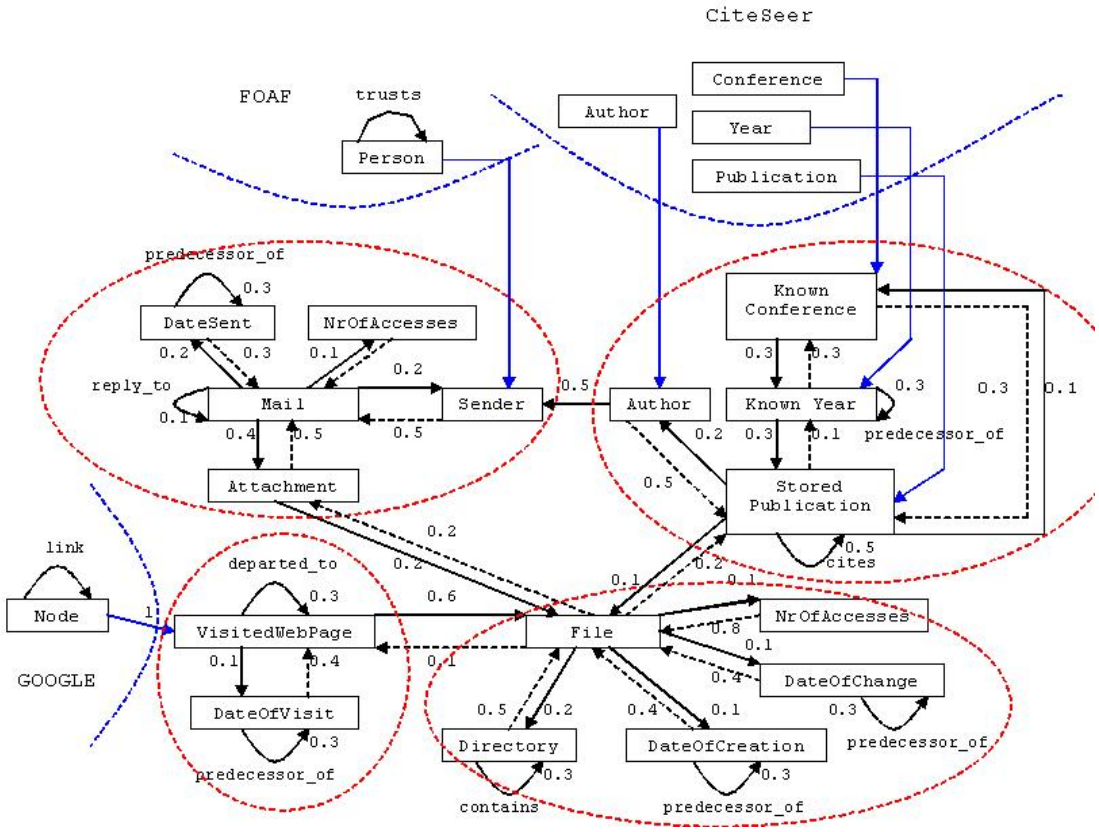

Figure 3.7: Contextual Authority Transfer Schema.

Figure 3.7 depicts our context ontology enriched with the appropriate authority transfer annotations. For example the authority of an email is influenced by the sender of the email, its attachment, the number of times that email was accessed, the date when it was sent and the email to which it was replied. Consequently, if an email is important, its sender might be an important person, its attachment an important one, just as the date when the email was sent and the previous email in the thread hierarchy. Given the fact that semantically based schema graphs are bi-directional, we split every edge in two parts, one for each direction. This is because contextual authority in particular flows in both directions: if we know that a 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 then calculated utilizing the regular PageRank formula, as applied onto the graph resulted from instantiating our ontology.

Personalization and Beyond. For the computation of authority transfer, we can also use PageRank's biasing vector in order to include additional external ranking sources into the algorithm, such as for example Google page scores, CiteSeer citation counts, or social network reputation values.

These Desktop specific rankings can already be seen as personalized, since they are specific to the data within each user's Personal Information Repository. However, the same PageRank personalization vector can be used to further bias the rankings onto some given contexts deemed more interesting by the user. At a lower granularity level, different authority transfer weights will express different preferences of the user, translating again into personalized rankings. The important requirement for doing this successfully is that we include in each user's ontology all concepts influencing her ranking function. For example if we consider a publication important because it was written by an important author, we have to represent that in the ontology. Similarly, if the importance of our digital photographies is heavily influenced by the event or the location where they were taken, then both of them must be included as classes in the context ontology.

3.3.3 Experiments

Experimental Setup

We evaluated our algorithms by conducting a small scale user study. Colleagues of ours provided a set of their locally indexed publications, some of which they received as attachments to emails (thus containing rich contextual metadata associated to them from the specific email fields). Then, each subject defined her *own* queries, related to their activities, and performed search over the above mentioned reduced images of their Desktops. In total, 30 queries were issued. The average query length was 2.17 keywords, which is slightly more than the average of 1.7 keywords reported in other larger scale studies (see for example [86]). Generally, the more specific the test queries are, the more difficult it is to improve over basic textual information retrieval measures such as TFxIDF. Thus, having an average query length a bit higher than usual can only increase the quality of our conclusions.

Our entire system is built as an extension to the open source Beagle Desktop search engine. For comparison purposes, we sent each of the above mentioned queries to three systems: (1) the original Beagle, whose output is selected and sorted using

solely TFxIDF, (2) an intermediate version of our system, Beagle^{++}, enhanced only with activity based metadata (using the same TFxIDF measure for ordering its output, but giving more importance to those results having also metadata associated to them), and (3) the current Beagle^{++}, containing enhancements for both metadata support and Desktop ranking. We combined the regular textual ranking with the link analysis based one using the following formula:

$$R'(a) = R(a) \cdot TFxIDF(a), \tag{3.1}$$

where $R(a)$ is the Desktop rank of resource a, and $TFxIDF(a)$ is automatically computed by Beagle. Thus, a search hit will have high score if it has both a high rank and a high TFxIDF score. Finally, for every query and every system, each user rated the top 5 output results using grades from 0 to 1, as follows: 0 for an irrelevant result, 0.5 for a relevant one, and 1 for highly relevant one.

Methodology

We used the ratings of our subjects to compute average precision and recall values at each output rank [17]. In general, precision measures the ability of an (information retrieval) system to return *only* relevant results. It is defined as:

$$\text{Precision} = \frac{\text{Number of } \textit{Relevant} \text{ Returned Results}}{\text{Number of Returned Results}} \tag{3.2}$$

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

$$\text{Recall} = \frac{\text{Number of Relevant Returned Results}}{\text{Total Number of Relevant Results Available in the Entire System}} \tag{3.3}$$

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

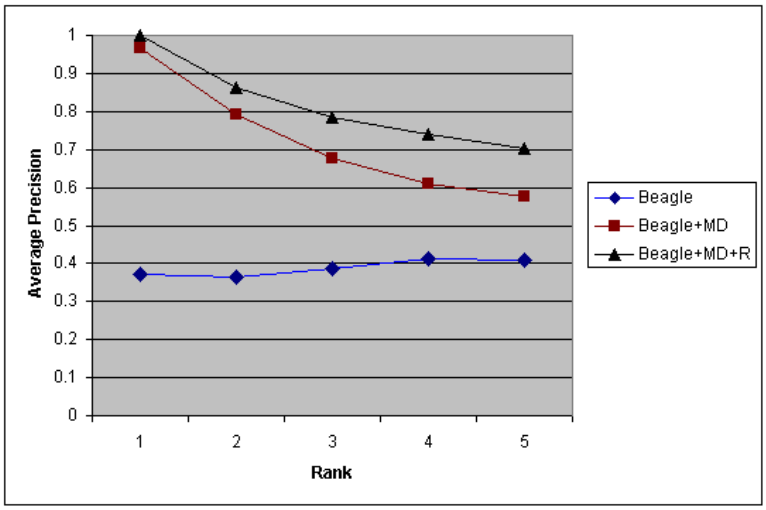

Figure 3.8: Average Precision Results.

is increased, recall is also increasing (the denominator remains the same, while the numerator has the possibility to increase), whereas precision is decreasing (because most of the relevant results should be at the very top of the list).

Another important aspect is calculating the total number of available relevant results. For search engines, including Desktop ones, an approximation must be used, as the datasets they cope with are too large. Here we consider this amount to be equal to the total number of (unique) relevant results returned by the three systems we investigated. For every query, each system returned 5 results, 15 in total. Thus, the minimum possible total number of relevant results is 0 and the maximum is 15. Similarly, the maximum number of relevant results a system can return is 5 (since it only outputs 5 results), indicating that the recall will not necessarily be 1 when restricting the computation to rank 5. This version of recall is called *relative recall* [17].

Results and Discussion

As the main purpose of our experimental analysis was to produce an estimate of each system's performance, we averaged the precision values at each rank from one to five for all 30 queries submitted by our experts. Note that this gave us a weighted precision, as we considered both relevant (i.e., scored 0.5) and highly relevant results (i.e., scored 1). The results obtained are depicted in Figure 3.8. We first notice that the current Beagle Desktop Search is rather poor, containing more qualitative results towards rank 4 to 5, rather than at the top of the result list. This is in fact explainable, since Beagle only uses TFxIDF to rank

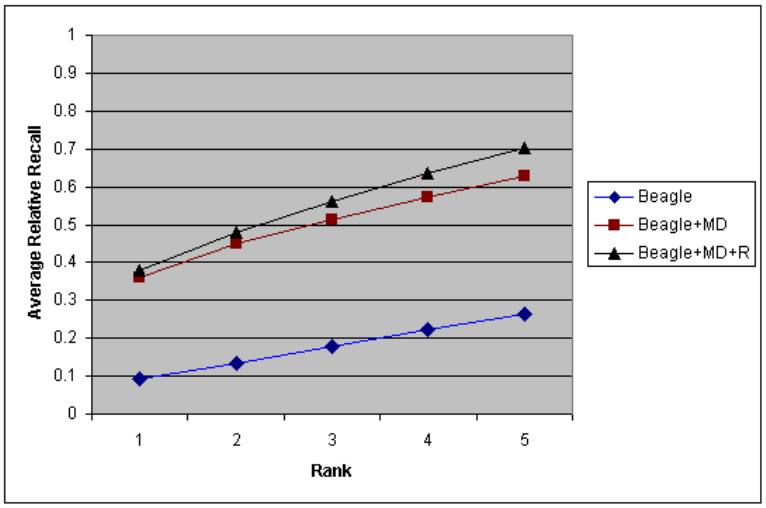

Figure 3.9: Average Relative Recall Results.

its results, thus missing any kind of global importance measure for the Desktop resources. On the contrary, our first prototype, consisting of Beagle enhanced with metadata annotations, already performs very well. An important reason for its high improvement is that metadata are mostly generated for those resources with high importance to the user, whereas the other automatically installed files (e.g., help files) are not associated with metadata, and thus ranked lower. Finally, the precision values are even higher for our second prototype, which adds our Desktop ranking algorithm to the metadata extended version of Beagle. Clearly ranking pushes our resources of interest more towards the top of the list, yielding even higher Desktop search output quality.

In the second part of the evaluation, we drew similar conclusions with respect to the average recall values (depicted in Figure 3.9): The recall of Beagle is very low, whereas that of our prototypes is almost three times better (owing to the additional information available as metadata). The difference between our two prototypes is relatively small, which is correct, since recall analyzes the amount of good results returned, and both our systems yield relevant results. We thus conclude that enhancing Beagle with metadata annotations significantly increases its recall (as metadata usually represents additional, highly relevant text associated to each Desktop file), whereas adding Desktop ranking further contributes with a visible improvement in terms of precision.

3.4 Ranking by Tracking All User Actions

In the previous section we proposed the first link analysis based algorithm which ranks resources within Personal Information Repositories. It was based on a set of heuristics defining several strict user activity patterns to generate relationships between Desktop items. Unfortunately, though they indeed connect highly related documents, these heuristics yield a rather sparse Desktop linkage matrix, leaving out many important resources from the computation. This section will propose and empirically investigate the opposite approach: *All* user activities will be taken into consideration when building the personal link structure. We will first introduce this new general heuristic in more detail, and then we will analyze both some Desktop specific behavioral patterns and their effect upon the defined ranking algorithm.

3.4.1 Generic Usage Analysis Based Ranking

Exploiting Resource Accesses to Generate Ranks. Our previous algorithm created links between Desktop resources only when a very specific Desktop usage activity was encountered (e.g., the attachment of an email was saved as a file, or a Web page was stored locally, etc.). We now address the problem from the other perspective and suppose that 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. This basically generalizes the algorithm introduced in Section 3.3.

We thus 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 less 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.

After a short period of time a reputation metric can be computed over the graph resulted from this usage analysis process. There exist several applicable metrics, the most common one being again PageRank. On the one hand, it has the advantage of propagating the inferred semantic similarities (connections), i.e., if there is a link between resources a and b, as well as an additional link between resources b and c, then with a relatively high probability we should also have a

connection between a and c. On the other hand, PageRank also implies a small additional computational overhead, which is not necessary for a simpler, yet more naïve metric, *in-link count*. According to this latter approach, the files accessed more often get a higher ranking. However, our experiments from Section 3.4.2 will show that although it does indeed yield a clear improvement over simple TFxIDF, file access counting is also significantly less effective than PageRank.

Another aspect that needs to be analyzed is the type of links residing on the PC Desktop. Since we are now dealing with a generic analysis, we use directed links for each sequence $a \rightarrow b$, as when file b is relevant for file a, it does not necessarily mean that the reversed is true as well. Imagine for example that b is a general report we are regularly appending, whereas a is the article we are writing. Clearly b is more relevant for a, than a is for b. This yields the following algorithm:

Algorithm 3.4.1.1. Ranking Desktop Items.

Pre-processing:
1: **Let** A be an empty link structure.
2: **Repeat** for ever
3: **If** (File a is accessed at time t_a, File b at time t_b) AND $(t_a - t_b < \epsilon)$,
4: **Then** Add the link $a \rightarrow b$ to A.

Ranking:
1: **Let** A' be an additional, empty link structure.
2: **For** each resource i
3: **For** each resource j linked to i
4: **If** $(\#Links(i \rightarrow j) > T)$ in A
5: **Then** Add one link $i \rightarrow j$ to A'.
6: **Run** PageRank using A' as underlying link structure.

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). Finally, in order to address a broad scope of possible ranking algorithms, we also experimented with more trivial reputation measures, namely (1) frequency of accesses and (2) total access time.

Exploiting Resource Naming and Content to Generate Ranks. There exists a plethora of other generic cues for inferring links between personal resources[10]. For example the *files stored within the same directory* have to some extent something in common, especially for filers, i.e., users that organize their personal data into carefully selected hierarchies [165, 20, 156]. Similarly, *files having the same file name* (ignoring the path) are in many times semantically related. In this case however, each name should not consist exclusively of stopwords. More, for this second additional heuristic we had to utilize an extended stopword list, which also includes several very common file name words, such as "index", or "readme". In total, we appended 48 such words to the original list. We also note that both these above mentioned approaches favor lower sets: If all files within such a set (e.g., all files residing in the same directory) are linked to each other, then the stationary probability of the Markov chain associated to this Desktop linkage graph is higher for the files residing in a smaller set. This is in fact correct, since for example a directory storing 10 items has most probably been created manually, thus containing files that are to some extent related, whereas a directory storing 1,000 items has in most of the situations been generated automatically.

A third broad source of linkage information is *file type*. There is clearly a connection between the resources sharing the same type, even though it is a very small one. Unfortunately, each such category will nowadays be filled with up to several thousands of items (e.g., JPG images), thus making this heuristic difficult to integrate into the ranking scheme. A more reliable approach is to *use text similarity to generate links between very similar Desktop resources*. Likewise, if *the same entity appears in several Desktop resources* (e.g., Hannover appears both as the name of a folder with pictures and as the subject of an email), then we argue that some kind of a semantic connection exists between the two resources. Finally, we note that users should be allowed to manually create links as well, possibly having a much higher weight associated to these special links.

Practical Issues. Several special cases might arise when applying usage analysis for Desktop search. First, the log file capturing usage history should persist over system updates in order to preserve the rich linkage information. In our experiments, we collected only about 80 KB of log data over 2 months. Second, more important, what if the user looks for a file she stored 5 years ago, when she had no Desktop search application installed? We propose several solutions to this:

1. The naïve approach is to simply enable ranking based exclusively on TFx-IDF. However, much better results can be obtained by incorporating contextual information within the ranking scheme.

[10]Note that these additional heuristics follow the general approach taken in this section, i.e., they are applicable to *all* personal files, rather than a set of narrowly specified ones.

2. We therefore propose a more complex query term weighting scheme, such as BM25 [132]. Teevan et al. [204] have recently proposed an application of this metric to personalize Web search based on Desktop content. In our approach, their method must be adapted to personalize *Desktop* search based on a specific subset of PIR, represented for example by the files with a specific path or date range.
3. If the user remembers the approximate moment in time when she accessed the sought item, then this date represents a useful additional context based vertical ranking measure. For example, if the user remembers having used the target file around year 1998, the additional importance measure is represented by the normalized positive time difference between mid-1998 and the date of each output result.
4. If no contextual information is available, we propose to infer it through a relevance feedback process, in which the user first searches the Desktop using TFxIDF exclusively, and then selects one or several (relatively) relevant results, which are then used to extract a context (e.g., date) or to propose expansions to the user query.

Comparison to the Web model. Clearly, unlike in the Web, most of the Desktop search queries are navigational: users just want to locate something they know their stored before. So, are some Desktop files more important than others, or are they all approximately equally important? We argue that, *as in the Web*, some Desktop resources are much more important than others, and thus users will most of the time be seeking only for these highly important items. For example, one year after some project was closed, a log file inspected by the researcher 400 times during an experiment will definitely be less important than the project report which was probably accessed only 100 times. Therefore, contextual information, though very important, is not sufficient in effectively locating Desktop items. More complex importance measures are thus needed in order to exploit user's activity patterns, her local Desktop organization, etc. either within a set of targeted scenarios, as in Section 3.3, or in a generalized approach, as described in this section.

3.4.2 Experiments

Experimental Setup

We evaluated the utility of our algorithms within three different environments: our laboratory (with researchers in different computer science areas and education), a partner laboratory with slightly different computer science interests, and the

architecture department of our university. The last location was especially chosen to give us an insight from persons with very different activities and requirements. In total, 11 persons installed our logging tool and worked normally on their Desktops for 2 months[11]. Then, during the subsequent 3 weeks, they performed several Desktop searches related to their regular activities[12], and graded each top 10 result of each algorithm with a score ranging from 1 to 5, 1 defining a very poor result with respect to their Desktop data and expectations, and 5 a very good one. This is in fact a Weighted P@10 [17], i.e., precision at the first 10 results. For every query, we shuffled the top ten URIs output by each of our algorithms, 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 the subjects had to evaluate about 30 Desktop documents (i.e., the reunion of the outputs of all approaches we investigated). In total, 84 queries had been issued and about 2,500 documents were evaluated.

For the link based ranking algorithms (recall that for the sake of completeness we have also evaluated some time based ranking heuristics) we set the parameter ϵ to four times the average break time of the user. We have also attempted to set it to one hour, and eight times the average break time of the user, but manual inspection showed these values to yield less accurate usage sessions. Although much more complex techniques for computing usage session times do exist (e.g., exploiting mouse clicks or movements, scrollbar activities, keyboard activities, document printing, etc. [74, 211]), we think this heuristic suffices for proving our hypothesis, i.e., usage analysis based ranking improves over simple textual retrieval approaches.

In the following we will first present an analysis of this experiment focused on identifying some general usage patterns and on investigating the behavior of our ranking algorithms. Afterwards we will proceed to the qualitative analysis of the search output produced by each approach.

Analyzing Usage Patterns

Our ultimate goal is to infer links from Desktop resources that have been accessed in a sequence. Yet this is not a straightforward task. Several persons might have quite different usage behavior strategies, thus making it very difficult to

[11]The logger was implemented using a hook that caught all manual file open / create / save system calls.

[12]The only requirement we made here was to perform at least 5 queries, but almost every subject provided more. In all cases, we collected the *average* rating per algorithm for each person.

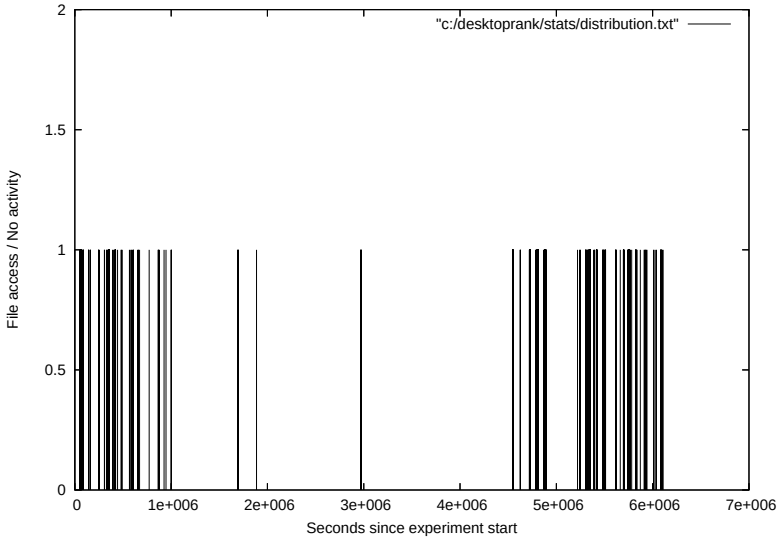

Figure 3.10: Two months distribution of manual accesses to desktop items.

distinguish usage sessions from each other. Moreover, this problem could even occur with the same person, at two different moments in time. We thus first analyzed the file access patterns of (seven of) our subjects. All of them manifested rather similar characteristics on the long term. We depict in Figure 3.10 one user activity captured over a period of two months. Notice that on such a long term the daily accesses are rather easy to distinguish: Each bar represents one file access. When several accesses occur at small intervals, their associated bars are merged into a thicker one. Also, longer breaks have been generated during week-ends, and the three very large pauses represent vacation periods. But what happens with the activity performed during a single day? A general example is presented in Figure 3.11. There are two access intensive working sessions in the morning, and *only one* session in the afternoon. In general, we distinguished two broad types of desktop activity: *Working* (e.g., reading an article, writing a program, etc.), which usually results in a relatively small file access frequency, and *Browsing* (e.g., reading emails, surfing the Web, etc.), when much more resources are opened in a short amount of time, in many cases only for reading. We believe these results could be used in a separate stream of research in order to find more accurate definitions for the parameter ϵ which delimits user sessions from each other.

Having identified the file access patterns, we then investigated the distributions of file access frequency and total file access time, as they represent a good indicator of how the final ranking distributions will look like. The former distribution has also been investigated by Dumais et al. [86], obtaining similar results. However, they only looked at the resources that have been accessed at least once, whereas we

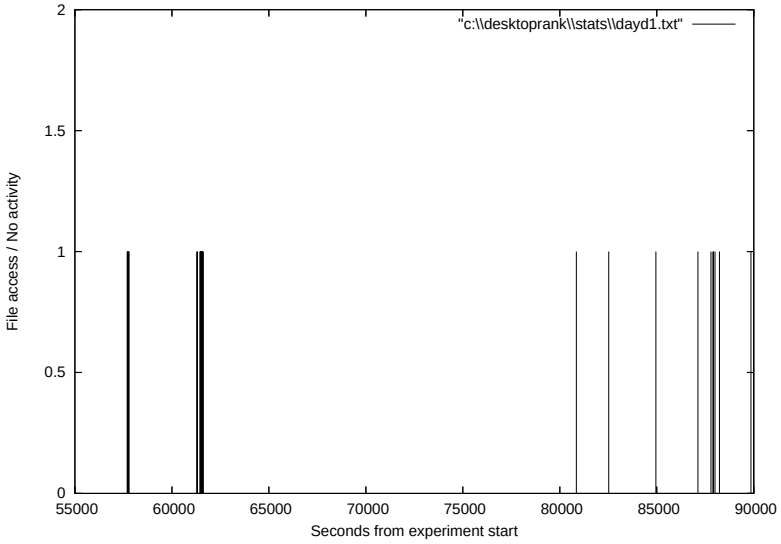

Figure 3.11: One day distribution of manual accesses to desktop items.

considered all Desktop items in our analysis. This helped us obtain an additional interesting result, namely that only about 2% of the Desktop indexable items are actually manually accessed by the user. This is most probably because of various program documentations (especially when in HTML format), locally stored mailing list archives, etc. We think this finding further supports the idea of exploiting usage information in ranking desktop search results, as current textual measures (i.e., TFxIDF) many times output high scores for such automatically deployed documents that have never been touched by the user. We depict a sample visit frequency distribution in Figure 3.12. For all our testers, this distribution followed a power law (i.e., $f(x) = c \cdot 1/x^{\gamma}$) with very low values for the γ exponent, ranging from -0.26 to -0.38. Similarly, we depict the distribution of total time spent accessing each file (i.e., reading, writing, etc.) in Figure 3.13. This distribution can be tailored by a power law with an exponential cut-off, as in the following formula:

$$f(x) = c \cdot \frac{1}{x^{\gamma}} \cdot e^{\frac{-x}{z_c}} \tag{3.4}$$

The additional inverse exponential term is only used to ensure a faster decreasing value of f, z_c being a parameter. Again, we obtained very low values for γ, residing around 0.18.

Ranking analysis

We then analyzed how our algorithms perform, in order to further tune their parameters and to investigate whether the non-usage analysis heuristics do indeed

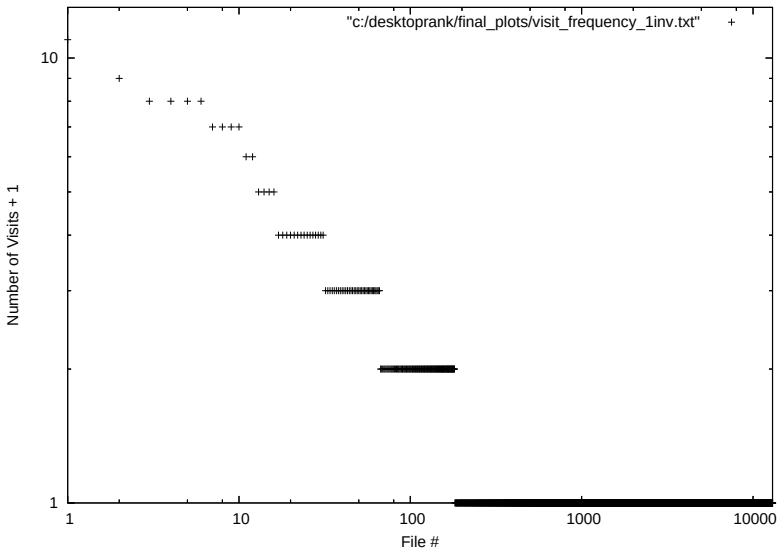

Figure 3.12: Frequency distribution of number of manual accesses to desktop items.

make a difference in the overall rankings. We thus defined and analyzed the following 17 algorithms:

- **T1**: Algorithm 3.4.1.1 with $T = 1$.
- **T1Dir**: "T1" enriched with additional links created as complete subgraphs with the files residing in every Desktop directory (i.e., all the files in a directory point to each other).
- **T1DirFnames**: "T1Dir" with further additional links created as complete subgraphs with the resources having the same file name (i.e., all items with the same file name point to each other, provided that the file name does not consist exclusively of stopwords).
- **T1Fnames**: "T1" enriched with the links between resources with identical file names as in the previous algorithm[13]. This was necessary to inspect the specific contribution of directories and file names respectively to the overall ranking scheme.
- **T1x3Dir**: Same as "T1Dir", but with the links inferred from usage analysis being three times more important than those inferred from the directory structure.
- **T1x3DirFnames**: Same as above, but also including the links provided by identical file names.

[13]For emails, this corresponded to having the same subject, eventually with "Re:" or "Fwd:" inserted in the beginning.

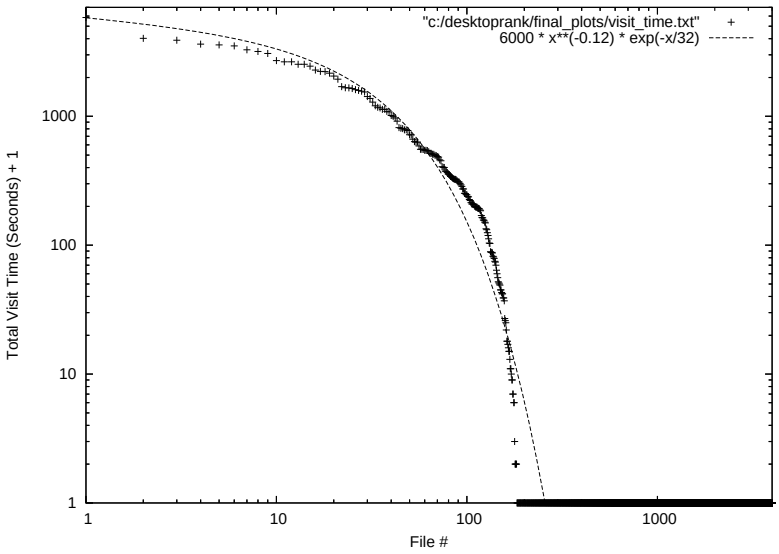

Figure 3.13: Distribution of time spent while manually accessing desktop items.

- **T1x3Fnames**: Same as "T1x3Dir", but using the file name heuristic instead of the directory one.
- **T2**: Algorithm 3.4.1.1 with $T = 2$.
- **T3**: Algorithm 3.4.1.1 with $T = 3$.
- **VisitFreq**: Ranking by access frequency.
- **1HourGap**: Ranking by total amount of time spent on accessing each resource, with sessions delimited by one hour of inactivity.
- **4xAvgGap**: Ranking by total access time, with sessions delimited by a period of inactivity longer than four times the average break time of the user.
- **8xAvgGap**: Same as above, but with sessions bounded by a period of inactivity longer than eight times the average break time of the user.
- **Weighted**: Algorithm 3.4.1.1 directly using the matrix A, instead of A', i.e., with links weighted by the number of times they occurred.
- **WeightedDir**: Algorithm "Weighted" enriched with links between the files stored within the same directory.
- **WeightedDirFnames**: The previous algorithm with a link structure extended with connections between files with identical names.
- **WeightedFnames**: Same as above, but without the links generated by exploiting the Desktop directory structure.

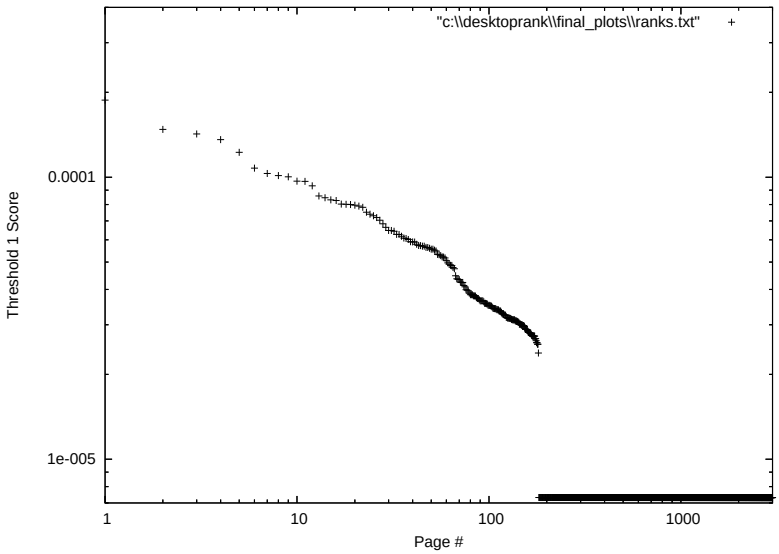

Figure 3.14: Distribution of scores for the "T1" algorithm.

Since in-link count is almost identical to file access count (frequency), we only experimented with the latter measure. The only difference between these two measures is that in-link count will result in lower page scores when a threshold higher than one is used to filter-out the links (see also Algorithm 3.4.1.1).

We analyzed two aspects at this stage: First, it was important to inspect the final distribution of rankings, as this indicates how Desktop search output looks like when using these algorithms. In *all* cases the resource rankings exhibits a distribution very well shaped by a power law: Figure 3.14 plots the output rankings for algorithm "T1", and Figure 3.15 depicts the output when both directory and file name heuristics were added (in this latter case we notice a strong exponential cut-off towards the end, for the files that benefited less from the link enhancement techniques).

The second aspect to analyze was whether there is a difference between these heuristics. For this purpose we used a variant of Kendall's τ measure of similarity between two ranking vectors [139], which resulted in a similarity score falling within [-1,1].

Three of our testers (one from each location) were specifically asked to extensively use our tool. When they reached 40 queries each, we applied the Kendall measure on their complete output, as returned by each algorithm. The results are illustrated in Table 3.1. After analyzing them, we drew the following conclusions:

- The heuristics to link the resources residing within the same directory, or the resources with identical file names did result in a rather different query

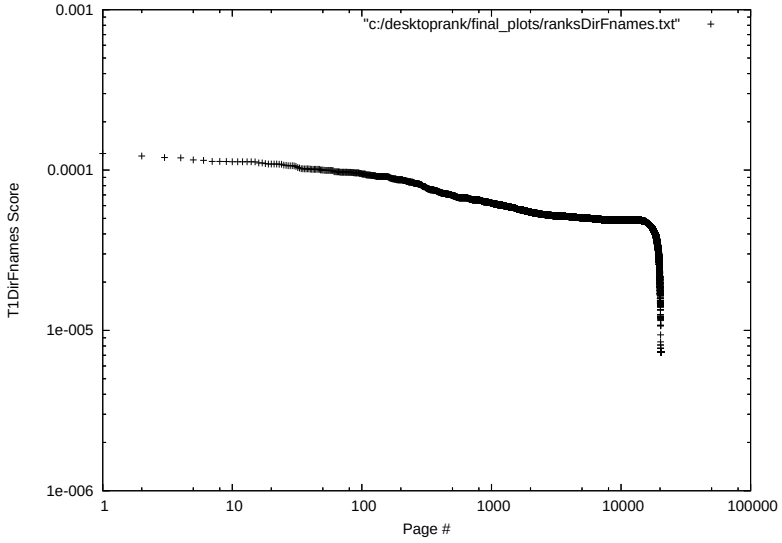

Figure 3.15: Distribution of scores for the "T1DirFnames" algorithm.

output.

- The approaches "T1x3Dir", "T1x3DirFnames" and "T1x3Fnames" did not yield a significant difference in the results.
- The output of "T2" and "T3" was very similar, indicating that a threshold higher than 2 is not necessary for Algorithm 3.4.1.1.
- "4xAvgGap" and "8xAvgGap" performed very similar to each other.
- "Weighted" output was very close to "T1".
- Finally, when "Weighted" was combined with directory or file name information, we obtained almost identical outcomes as when we used "T1" with these heuristics.

As a rule of thumb, we considered similar all algorithm pairs with a Kendall τ score above 0.5, and therefore removed one of them from the search quality evaluation. Exceptions were "Weighted" and "VisitFreq" (both very similar to "T1") in order to have at least one representative of their underlying heuristics as well.

Search quality analysis

After the previous analysis, we kept 8 algorithms for precision evaluation: "T1", "T1Dir", "T1DirFnames", "T1Fnames", "T2", "VisitFreq", "4xAvgGap" and "Weighted". Even though they do not incorporate any textual information, we still started with ranking Desktop search results only according to these measures, in order to see the impact of usage analysis on Desktop ranking. The average

Algorithm	T1	T1Dir	T1DirFnames	T1Fnames	T1x3Dir	T1x3DirFnames	T1x3Fnames	T2	T3	VisitFreq	1HourGap	4xAvgGap	8xAvgGap	Weighted	WeightedDir	WeightedDirFnames	WeightedFnames
Threshold 1	1																
T1Dir	0.2	1															
T1Dir-Fnames	0.2	0.5	1														
T1Fnames	0.2	0.2	0.4	1													
T1x3Dir	0.3	0.9	0.5	0.2	1												
T1x3Dir-Fnames	0.2	0.5	0.8	0.4	0.5	1											
T1x3Fnames	0.2	0.2	0.4	0.9	0.2	0.4	1										
Threshold 2	0.2	0	-0.2	0	0	-0.2	0	1									
Threshold 3	0	-0.1	-0.3	-0.1	-0.1	-0.3	-0.1	0.6	1								
VisitFreq	0.7	0.2	0.2	0.3	0.3	0.2	0.3	0.3	0.1	1							
1HourGap	0.5	0.2	0.1	0.2	0.1	0.1	0.2	0.2	0	0.4	1						
4xAvgGap	0.4	0.3	0.2	0.2	0.3	0.2	0.2	0.2	0	0.4	0.3	1					
8xAvgGap	0.5	0.3	0.2	0.2	0.3	0.2	0.2	0.2	0	0.5	0.5	0.7	1				
Weighted	0.8	0.2	0.2	0.2	0.3	0.2	0.2	0.2	0	0.6	0.5	0.5	0.5	1			
WeightedDir	0.2	0.9	0.5	0.2	0.9	0.5	0.2	0	-0.1	0.2	0.1	0.3	0.3	0.2	1		
Weighted-DirFnames	0.2	0.5	0.9	0.3	0.5	0.8	0.4	-0.2	-0.3	0.2	0.1	0.2	0.2	0.2	0.5	1	
Weighted-Fnames	0.3	0.2	0.4	0.8	0.3	0.4	0.8	0	-0.1	0.3	0.3	0.3	0.3	0.3	0.2	0.4	1

Table 3.1: Kendall similarity for the Desktop ranking algorithms (average over 120 queries from 3 users).

results are summarized in the second column of Table 3.2. As we can see, all measures performed worse than TFxIDF (we used Lucene[14] together with an implementation of Porter's stemmer to select the query hits, as well as to compute the TFxIDF values[15]), but only at a small difference. This indicates that users do issue a good amount of their Desktop queries on aspects related to their relatively recent, or even current work. Also, as the "T2" algorithm does not improve over "T1", it is therefore sufficient to use Algorithm 3.4.1.1 with a threshold $T = 1$ in order to effectively catch the important Desktop documents. This is explainable,

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

[15]Note that even though our Desktop search system, Beagle^{++}, is also based on a Linux specific .Net implementation of Lucene, we decided to move this evaluation outside of its context, in order to allow for a broader range of subjects, i.e., running Java on both Linux and Windows.

since a threshold $T = 2$ would only downgrade files that were accessed only once, which have a relatively low score anyway compared to the other more frequently touched resources.

Finally we investigated how our algorithms perform within a realistic Desktop search scenario, i.e., combined with term frequency information. We used the following formula:

$$Score(file) = NormalizedScore(file) \cdot NormalizedVSMScore(file, query)$$

The VSM score is computed using the Vector Space Model and both scores are normalized to fall within [0,1] for a given query[16]. The resulted average gradings are presented in the third column of Table 3.2. We notice that in this approach, *all* measures outperform TFxIDF in terms of weighted precision at the top 10 results, and most of them do that at a statistically significant difference (see column 4 of Table 3.2 for the p values with respect to each metric).

The usage analysis based PageRank ("T1") is clearly improving over regular TFxIDF ranking. As for the additional heuristics evaluated, connecting items with similar file name or residing in the same directory, they yielded a significant improvement only when both of them have been used. This is because when used by themselves, these heuristics tend to bias the results away from the usage analysis information, which is the most important by far. When used together, they add links in a more uniform manner, thus including the information delivered by each additional heuristic, while also keeping the main bias on usage analysis. Finally, the simpler usage analysis metrics we investigated (e.g., ranking by frequency or by total access time) did indeed improve over TFxIDF as well, but with a lower impact than the Algorithm 3.4.1.1 enriched with directory and file name information. We conclude that with TFxIDF in place, usage analysis significantly improves Desktop search output rankings and it can be further enhanced by linking resources from the same directory and with identical file names.

The final results are also illustrated in Figure 3.16, in order to make the improvement provided by our algorithms also visible at a graphical level. The horizontal line residing at level 3.09 represents the performance of TFxIDF; the red bars depict the average grading of the algorithms combining TFxIDF with our approaches, and the blue ones depict the average grading obtained when using only our usage analysis algorithms to order Desktop search output.

Observation. We have showed our algorithms to provide significant improvements over regular TFxIDF search. Yet since they tend bias the results very

[16]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.

Algorithm	Weighted P@10 (Usg. An.)	Weighted P@10 (Combined)	Signif. for Combined versus TFxIDF
T1 · TFxIDF	3.04	3.34	$p = 0.003$
T1Dir · TFxIDF	3.02	3.36	$p < 0.001$
T1DirFnames · TFxIDF	2.99	3.42	$p \ll 0.001$
T1Fnames · TFxIDF	2.97	3.26	$p = 0.064$
T2 · TFxIDF	2.85	3.13	$p = 0.311$
VisitFreq · TFxIDF	2.98	3.23	$p = 0.141$
4xAvgGap · TFxIDF	2.94	3.09	$p = 0.494$
Weighted · TFxIDF	3.07	3.30	$p = 0.012$
TFxIDF	3.09	3.09	

Table 3.2: Average grading for the usage analysis algorithms with and without a combination with TFxIDF, together with tests on the statistical significance of the improvement the latter ones bring over regular TFxIDF.

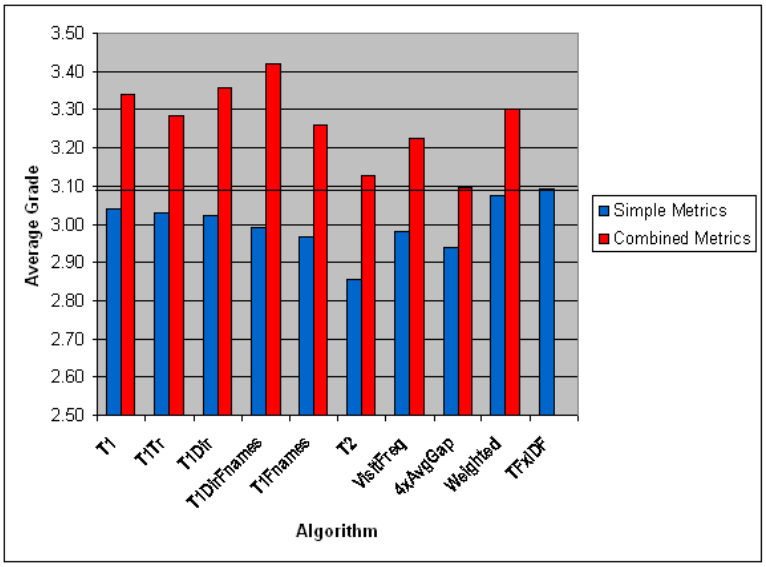

Figure 3.16: Average grading for the usage analysis algorithms.

much towards the previously accessed files, wouldn't we get similar performance simply by boosting these visited files within the Desktop search output (i.e., without using any link analysis approach)? The answer is *no* and can be probed by carefully analyzing the results from Table 3.2. The "VisitFreq" algorithm is in fact an implementation of this naïve heuristic: it orders the search results based on the frequency with which they were visited; if this is 0, then TFxIDF is used to sort the output. And as we can see from Table 3.2, VisitFreq performs significantly worse compared to our complete link analysis approach, T1DirFnames (3.23 versus 3.42, a difference which is also statistically significant with $p = 0.043$), as well as to T1, which only considers the creation of access based links (3.23 versus 3.34,

with $p = 0.098$). Therefore, a more complex measure of importance for Desktop items such as T1DirFnames is indeed necessary in order to also incorporate the *order* in which resources have been touched, as well as any other potential Desktop linkage heuristics.

3.5 Discussion

Currently there are quite several personal information systems managing PC Desktop resources. However, all of them have focused on seeking solutions to find previously stored items in a faster way. In this chapter we argued that Personal Information Repositories have grown considerably in size, and thus they are emerging as a potentially problematic environment in terms of Data Management. More specifically, we argued that all previous search based solutions to locate information at the PC Desktop level are insufficient for the current scalability requirements. These approaches already yield several hundreds of query results, which cannot be successfully ordered by using textual retrieval measures exclusively. To solve this problem we proposed to introduce *link analysis ranking* for Desktop items and investigated in detail two major approaches to achieve this goal:

- First, we exploited contextual analysis of specific user actions to build the PIR link structure and compute "local reputation" values for personal items. This approach does indeed an excellent job at ranking Desktop search output. However, each manually defined activity context brings only a relatively small amount of links into the local graph connecting user's resources. Thus, a lot of design and programming work is needed in order to achieve both quality and coverage.
- In the second part we generalized this idea by including all user actions into the ranking algorithm. This approach was clearly much simpler, while also providing a visibly larger coverage of the personal items. Unfortunately, the qualitative improvements it brought were smaller than those of the previous technique, even though they were still improving over regular Desktop search at a statistically significant difference. This small quality decrease is because of the noisy connections it induces into the local link structure. As it is a general heuristic, it also captures false links, such as those generated due to frequent switches between Desktop activity contexts.

All in all, as we found out that people usually access only about 2% of their indexable items, we conclude that both our usage analysis based ranking algorithms are very suitable for a better Personal Information Management. In fact, our

user based experiments showed both techniques to significantly increase Desktop search quality.

Two issues still remain open: First, it would be interesting to define some better fully automatic graph trimming heuristics for our second approach, in order to keep the ranking coverage at a sufficiently high level, while also achieving an excellent output quality. Second, though less important, one might investigate other ranking extensions which include also non access based heuristics, thus addressing more local resources, even when these have never been opened by the user.

Chapter 4

Ranking for Spam Detection

4.1 Introduction

Spamming is the abuse of using electronic messaging systems to send unsolicited messages. Though its most widely recognized form is *Email spam*, the term is also applied to similar abuses in other media: Instant Messaging spam, Usenet newsgroup spam, Web search engine spam, Weblogs spam, and Mobile phone messaging spam.

In this chapter we tackle the two most important forms of spamming, namely Email and Web spam. Email Spam is a very profitable business, as it has no associated operating cost beyond the actual management of the recipient addresses. Thus, malicious merchants would send millions of Emails either promoting some underground products (for example Viagra medicines), or attempting to lure the addressee into fake businesses (quite spread is the model of the Nigerian businessmen who inherited a large amount of money, yet is in need of an account to ship them to), etc. Moreover, senders of these Emails are difficult to identify and to hold responsible for their mailings. Consequently, spammers (i.e., creators of spam) are numerous, and the volume of unsolicited mail has become extremely high. Unlike legitimate commercial Email, spam is generally sent without the explicit permission of the recipients, and frequently contains various tricks to bypass Email filtering. Even though current retail Email services offer the possibility to report any incurred spam in order to constantly update their anti-spam filters, this is and will remain a continuous fight (at least up to a certain moment), in which each party is devising new techniques, either to deceive or to fight its opponent.

A similar phenomenon can be observed for search engines as well, as they are indexing more and more content every day. If we also remember that upon searching this huge quantity of data, people usually view only the very few top answers returned for each query, it becomes highly important to provide these search results with the best quality possible. Alas, currently this is not an easy task. Spamdexing (a portmanteau of spamming and Web indexing) refers to the practice on the World Wide Web of maliciously modifying HTML pages in order to increase their chances of being placed high on search engine ranking lists, either because their text has been altered to contain some targeted words very frequently, or because many artificial Web hyperlinks have been added towards them, etc. Though more costly and more difficult to accomplish than its Email surrogate, Web spamming is also bringing enhanced profits, especially for sites dedicated to activities such as online gambling or porn, as long as they manage to rank high for a certain number of common Web search queries. This is because high ranking many times also implies high traffic, which consequently usually converts into a high revenue for the specific Web site.

This chapter proposes to exploit link analysis ranking solutions for both Email and Web spam detection. While this approach has been previously explored for the latter domain, it is rather new for the former one. We thus propose to link people across the social network created upon their exchanges of Emails: Once a person sends an Email to a recipient, a vote is automatically cast towards that destination individual. The amount of Emails one *receives*, as well as the "importance" of their senders, contribute to achieving a high rank within the Email social network. Since spammers generally only send Emails (i.e., cast votes in our system), their overall reputation scores will turn out to be rather low, being (at least partially) separated from the rest of the network. Moreover, using this Email ranking approach, one would be able to order its incoming Emails according to the global reputation of their sources. Abstracting from the application environment, it is easy to observe that this is in fact the same model as the one currently employed by the Web search engines. There, social interactions are described by the process of creating hyperlinks between pages, which are then regarded as votes within the reputation algorithms. Unfortunately, as this model has already been exploited for several years already within the Web environment, spammers have already invented new and increasingly more complex link reinforcement structures to boost their reputation scores. Thus, simple PageRank-like schemes are no longer sufficient for detecting malicious users in the World Wide Web. In the second part of this chapter we propose an improved technique to isolate spammers: We analyze the relationships between social groups in order to discover entities involved in irregular linkage structures. Once these have been identified, we simply remove them from the reputation algorithm. We test this technique onto a large collection

of Web pages, and show that link databases cleaned of such noisy information yield significantly better Web search results. Finally, while the social network based model we propose for ordering Email addresses is rather primitive, similar to the ones used in the Web about 8 years ago, we believe that if adopted, its development would be somewhat similar to that of the Web, and consequently all recent findings from the Web link analysis research would be fairly simple to adapt and apply for Emails as well.

The chapter is organized as follows: First, we give an overview of the existing spam detection and social network reputation metrics in Section 4.2. Then, we first discuss why and how these metrics could be applied to counter Email spam in Section 4.3. These being introduced, we transit in Section 4.4 towards some higher level link spam detection approaches. We deploy these solutions onto the currently more complex Web environment and analyze them using the very same social reputation model (i.e., PageRank). In the end of each of these two sections we conclude with a discussion of possible future research directions, as focused on the investigated medium, Email or Web.

4.2 Specific Background

Ranking has generally not been explicitly used to enforce spam detection in the literature. Especially for the Email application, most techniques focused on content and communication protocol extensions, rather than social network based reputation mechanisms. In this chapter we propose ranking as a viable solution for detecting malicious users in social environments. It has been already proved that social reputation mechanisms are good at distilling out qualitative subjects within global communities [172, 135]. In the same time they are also particularly stable and resistant in front of various attacks. Considering the fact that all media attacked by spammers are actually social media (i.e., Email, Web, Instant Messaging, etc.), we argue that utilizing social ranking algorithms could be very benefic in filtering the members of their underlying communities.

This section starts with a generic discussion about Email anti-spam approaches. Afterwards, we give an overview of general reputation algorithms for social networks, as they are built on top of basic solutions to isolate malicious users. Finally, in the last part we present how these ranking algorithms have been employed, as well as enhanced for spam detection in the World Wide Web.

4.2.1 Email Anti-Spam Approaches

Because of the high importance of the Email spam problem, many attempts to counter spam have been started in the past, including some law initiatives. Technical anti-spam approaches comprise one or several of the following basic solutions [175]:

- Content-based approaches
- Header-based approaches
- Protocol-based approaches
- Approaches based on sender authentication
- Approaches based on social networks

Content-based approaches [113] analyze the subject of an Email or the Email body for certain keywords (statically provided or dynamically learned using a Bayesian filter) or patterns that are typical for spam Emails (e.g., URLs with numeric IP addresses in the Email body). The advantage of content-based schemes is their ability to filter quite a high number of spam messages. For example, SpamAssassin can recognize 97% of the spam if an appropriately trained Bayesian filter is used together with the available static rules [128]. However, in contrast to the social network reputation models, content based approaches need to continuously adapt their set of static rules, as otherwise their high spam recognition rate will decrease.

Header-based approaches examine the headers of Email messages to detect spam. *Whitelist schemes* collect all Email addresses of known non-spammers in a whitelist to decrease the number of false positives from content-based schemes. In contrast, *blacklist schemes* store the IP addresses (Email addresses can be forged easily) of all known spammers and refuse to accept Emails from them. A manual creation of such lists is typically highly accurate but puts quite a high burden on the user to maintain it. PGP key servers could be considered a manually created global whitelist. An automatic creation can be realized, for instance based on previous results of a content-based filter as is done with the so-called *autowhitelists* in SpamAssassin. Both blacklists and whitelists are rather difficult to maintain, especially when faced with attacks from spammers who want to get their Email addresses on the list (whitelist) or off the list (blacklist). They are related to our Email spam detection approach in the sense of creating lists with trusted / malicious users. However, our algorithm is fully automatic, thus being also fairly easy to maintain.

Protocol-based approaches propose changes to the utilized Email protocol. *Challenge-response schemes* [175] require a manual effort to send the first Email

to a particular recipient. For example, the sender has to go to a certain Web page and activate the Email manually, which might involve answering a simple question (such as solving a simple mathematical equation). Afterwards, the sender will be added to the recipient's whitelist such that further Emails can be sent without the activation procedure. The activation task is considered too complex for spammers, who usually try to send millions of spam Emails at once. An automatic scheme is used in the *grey-listing* approach[1], where the receiving Email server requires each unknown sending Email server to resend the Email again later. Nevertheless, these techniques seem to impose a too high burden on honest users as well, since they have not been adopted on a wide scale. Also, as they build upon user interaction procedures, they are orthogonal to our spam detection solutions.

To prevent spammers from forging their identity (and allow for tracking them), several approaches for **sender authentication** [108] have been proposed. They basically add another entry to the DNS server, which announces the designated Email servers for a particular domain. A server can use a reverse lookup to verify if a received Email actually came from one of these Email servers. Sender authentication is a requirement for whitelist approaches (including ours), since otherwise spammers could just use well-known Email addresses in the "From:" line. Though it is already implemented by large Email providers (e.g., AOL, Yahoo!), it also requires further mechanisms, such as a blacklist or a whitelist. Without them, spammers could simply set up their own domains and DNS servers, thus easily circumventing the system.

Recent approaches have started to exploit social interactions for spam detection. Such **social network based approaches** construct a graph, whose vertices represent Email addresses. A directed edge is added between two nodes A and B, if A has sent an Email to B. Boykin and Roychowdhury [31] classify Email addresses based on the clustering coefficient of the graph subcomponent: For spammers, this coefficient is very low because they typically do not exchange Emails with each other. In contrast, the clustering coefficient of the subgraph representing the actual social network of a non-spammer (colleagues, friends, etc.) is rather high. The scheme can classify 53% of the Emails correctly as ham or spam, leaving the remainder for further examination by other approaches. This is similar to the algorithm we present in Section 4.3 in the sense that it uses link analysis for spam detection. However, we also exploit the power-law distribution of social contacts, thus being able to obtain a much more accurate classification ratio, as well as to order Email addresses by the social reputation of their senders. Finally, closest to our work is the paper of Golbeck and Hendler, who propose a typical spreading activation scheme to rank Email addresses, based on exchanges

[1] http://projects.puremagic.com/greylisting/

of reputation values [112]. They still achieve a spam detection ratio below ours. More important, the real-life applicability of their scheme is uncertain, as its attack resilience has not been verified at all.

4.2.2 Trust and Reputation in Social Networks

Trust and reputation algorithms have become increasingly popular to rank a set of items, such as people (social reputation) or Web pages (Web reputation), for example, when selling products in online auctions. Their main advantage is that most of them are designed for high attack resilience.

Social reputation schemes have been designed mostly for use over Peer-To-Peer networks [69, 68, 57]. However, they provide an useful insight into using link analysis ranking to construct reputation systems, as well as into identifying different attack scenarios. From this perspective, they are also very similar to our algorithms. The only significant difference is that they have been adapted for and deployed onto different application environments. Ziegler and Lausen [224] present a general categorization of trust metrics, as well as a fixed-point personalized trust algorithm inspired by spreading activation models. It can be viewed as an application of PageRank onto a sub-graph of the social network in order to compute user reputation scores. Richardson [181] builds a Web of trust asking each person to maintain trust values on a small number of other users. The algorithm presented is also based on a power iteration, but designed for an application within the context of the Semantic Web, composed of logical assertions. Finally, EigenTrust [135] is a pure fixed-point PageRank-like distributed computation of reputation values for Peer-To-Peer environments. This algorithm is also used in the MailTrust approach [146], an Email reputation metric highly similar to Basic MailRank, investigated into the physics research community slightly after our first report became public. MailTrust is still different from our approach in the sense that it does not investigate any user oriented Email filtering or importance ordering. Moreover, it builds upon a straightforward application of PageRank, which is much more sensible to malicious attacks than MailRank (note that in contrast, we bias PageRank onto the highly reputable members of each social community, thus making the gap between trustful and malicious users significantly larger).

4.2.3 Spam Detection in the World Wide Web

Detecting Spam on the Web. The more money are circulated within an environment, the more interest will spammers have to get a share of it. Naturally, this is also valid for the World Wide Web. This makes finding good solutions

for the Web spam detection problem not only important, but also difficult, as they need to deal with adversaries that continuously try to deceive search engine algorithms. As the Web search output is usually ordered using a *combination* of various techniques available to assess the quality of each result (e.g., PageRank, HITS, TFxIDF, etc.), spammers have devised specific schemes to circumvent each of these measures. Consequently, the search engines responded with detection or neutralization techniques, usually built on top of basic Web reputation algorithms similar to those outlined in the previous section. This caused the spammers to seek new rank boosting methods, and so on. Since our work is focused on identifying spam using the link structure describing various social media, we will present in this section only the most recent anti-spam techniques for link-based Web ranking algorithms. Whenever possible, we will compare these approaches with the algorithms we propose and describe in Section 4.4.

Ranking Based Approaches. The currently known types of artificial link structures which could boost the rank of one or more Web pages have been investigated by Gyögyi et al. [114]. They manually built toy-scale link farms (networks of pages densely connected to each other) or alliances of farms and calculated their impact upon the final rankings. We used their results to design some of our spam fighting algorithms.

The seminal article of Bharat and Henzinger [27] has indirectly addressed the problem of spam neutralization on the Web. Though inherently different from our approaches, the work provides a valuable insight into Web link spam: The authors discovered the existence of "mutually reinforcing relationships" and proposed to assign each edge (i.e., hyperlink) an authority weight of $1/k$ if there are k pages from one Web site pointing a single document from another site, as well as a hub weight of $1/l$ if a page from the first site is pointing to l documents residing all on the second site. Authors use this information to change HITS [143], the hub and authority ranking algorithm which we also described in Section 2.1.2. We believe their solution could be used to complement our spam detection approaches, both for Web pages / sites and for Emails, in which corporation wide domains can be seen as "sites". Later, Li et al. [161] also proposed an improved HITS algorithm to avoid its vulnerability to *small-in-large-out* situations, where one page has only a few in-links but many out-links. Nevertheless, their work focused only on this specific problem, thus not tackling spam detection per se.

Another important work is SALSA [158], where the "Tightly-Knit (TKC) Community Effect" was first discussed. The organization of pages into such a densely linked graph usually results in increasing their scores. The authors proposed a new link analysis algorithm which adopts two Markov chains for traversing the Web graph, one converging to the weighted in-degree of each page, for authority scores,

and the other converging to its weighted out-degree, for hub scores. The approach resembles popularity ranking, which was also investigated by Chakrabarti [46] and Borodin et al. [29]. However, it does not incorporate any iterative reinforcement and is still vulnerable to some forms of the TKC effect [184].

Zhang et al. [223] discovered that colluding users amplify their PageRank score with a value proportional to $Out(1/c)$, where c is the PageRank dampening factor[2]. Thus, they propose to calculate PageRank with a different c for each page p, automatically generated as a function of the correlation coefficient between $1/c$ and $PageRank(p)$ under different values for c. Their work is extended by Baeza-Yates et al. [15], who study how the PageRank increases under various collusion (i.e., nepotistic) topologies and prove this increase to be bounded by a value depending on the original PageRank of the colluding set and on the dampening factor.

BadRank[3] [216] is one of the techniques supposed to be used by search engines against link farms. It is an inverse PageRank, in which a page gets a high score if it points to many pages with high BadRank, as in the formula below:

$$BR(p) = c \cdot \sum_{q \in In(p)} \frac{BR(q)}{\|Out(q)\|} + (1-c) \cdot IB(p) \tag{4.1}$$

The exact expression of $IB(p)$ is not known, but it represents the initial BadRank value of page p as assigned by spam filters, etc. The algorithm is complementary to our approaches and the idea of propagating the badness score of a page could be implemented as an extension on top of the algorithms presented in Section 4.4.

TrustRank [116] proposes a rather similar approach, but focused on the good pages: In the first step, a set of high quality pages is selected and assigned a high trust; then, a biased version of PageRank is used to propagate these trust values along out-links throughout the entire Web. The algorithm is orthogonal to our approaches: Instead of seeking for good pages, we attempt to automatically identify and penalize malicious nodes (for Email spam) and links (for Web spam).

SpamRank [25] resembles an "opposite TrustRank": First, each page receives a penalty score proportional to the irregularity of the distribution of PageRank scores for its in-linking pages; then, Personalized PageRank is used to propagate the penalties in the graph. The advantage over TrustRank is that good pages cannot be marked as spam, and comes at a cost of higher time complexity. Our Web spam detection approach is similar with respect to penalizing bad pages, but we build our set of malicious candidates much faster, by identifying abnormal

[2]Recall the definition of PageRank from Equation 2.1: $PR(p) = c \cdot \sum_{q \in I(p)} \frac{PR(q)}{\|O(q)\|} + \frac{(1-c)}{\|V\|}$.

[3]http://en.efactory.de/e-pr0.shtml

link structures, instead of analyzing the distribution of PageRank scores for the in-linking pages of each page.

Wu and Davison [216] first mark a set of pages as bad, if the domains of n of their out-links match the domains of n of their in-links (i.e., they count the number of domains that link to and are linked by that page). Then, they extend this set with all pages pointing to at least m pages in the former set, and remove all links between pages marked as bad. In the end, new rankings are computed using the "cleaned" transition probability matrix. Their algorithm is not applicable for our Email approach, as it is quite common for a group of persons to exchange Emails without forming a malicious collective. However, it is complementary to our Web spam detection scheme, as it operates at the lower level of Web pages, instead of sites. In [217], the same authors build bipartite graphs of documents and their "complete hyperlinks"[4] in order to find link farms of pages sharing both anchor text and link targets (i.e., possibly automatically created duplicate links). Again, the algorithm makes a good complement for our Web scenario.

Finally, Becchetti et al. [23] compute Web page attributes by applying rank propagation and probabilistic link counting over the Web graph. They are thus able to estimate the number of supporters of each node in a graph. More interesting, they show how to truncate this value to only consider neighbors at a distance higher than d, which consequently enables the computation of PageRank without any link cycles of length smaller than d, most of which are usually artificially created, especially with very small values for d. The performance of this approach was then compared with many other Web spam classifiers in [22], reaching at most 80.4% accuracy when the available indicators were combined.

Other Approaches. While most Web spam detection research has concentrated directly on the link analysis algorithms used within current search engines, another significant stream of activity was dedicated to designing innovative third party solutions to detect such unwanted hyperlinks. These are specifically tailored to the characteristics of the Web content, and thus applicable only in this environment. Kumar et al. [149] used bipartite graphs to identify Web communities and marked as nepotistic those communities having several fans (i.e., pages contributing to the core of the bipartite graph with their out-links) residing on the same site. Roberts and Rosenthal [184] analyzed the number of *Web clusters* pointing to each target page in order to decrease the influence of TKCs. They proposed several methods to approximate these clusters, but they evaluated their approach only minimally. A rather different technique was employed in [10], where the authors presented a decision-rule classifier employing 16 connectivity features (e.g., average level of page in the site tree, etc.) to detect Web site functionality. They claimed to have

[4]Hyperlinks having the anchor text attached to them.

successfully used it to identify link spam rings as well, but no details are given about the importance of each feature for accomplishing this task.

Chakrabarti [45] proposed a finer grained model of the Web, in which pages are represented by their Document Object Models, with the resulted DOM trees being interconnected by regular hyperlinks. The method is able to counter "nepotistic clique attacks", but needs more input data than our Web algorithms, which are based exclusively on link analysis. Also, we are able to identify a lager group of link anomaly types.

Fetterly et al. [99] used statistical measures to identify potential spam pages. Most of the features they analyzed can be modeled by well known distributions, thus placing outliers in the position of potential spammers. After a manual inspection, the vast majority of them seemed to be spammers indeed. A related technique to detect spam pages is based on machine learning algorithms: Davison [79] used them on several features of URLs (e.g., similar titles, domains, etc.) in order to identify nepotistic links on the Web.

Finally, note that there exist also several types of *noisy* hyperlinks, which are not necessarily spam. The most common one is due to mirror hosts and can be eliminated using algorithms such as those proposed by Broder et al. [35] or Bharat et al. [26]. Also, navigational links are intended to facilitate browsing, rather than expressing votes of trust. One work indirectly related to this type of links is [91], where the authors defined Web documents as a "cohesive presentation of thought on a unifying subject" and proposed using these entities for information retrieval, instead of the regular Web pages. Their work is however orthogonal to ours, as they seek to identify the correct Web entities, whereas we propose solutions to remove spam items (i.e., links and nodes) from search engine databases.

4.3 Ranking for Email Spam Detection

We now turn our attention to the first and most common form of spam: Email spam. While scientific collaboration without Email is almost unthinkable, the tremendous increase of spam over the past years [108] has rendered Email communication without spam filtering almost impossible. Currently, spam Emails already outnumber non-spam ones, so-called 'ham Emails'. Existing spam filters such as SpamAssassin[5], SpamBouncer[6], or Mozilla Junk Mail Control[7] still exhibit a number of problems, which can be classified in two main categories:

[5] http://spamassassin.apache.org
[6] http://www.spambouncer.org
[7] http://www.mozilla.org/start/1.5/extra/using-junk-control.html

1. **Maintenance,** for both the initialization and the adaptation of the filter during operation, since all spam filters rely on a certain amount of input data to be maintained: Content-based filters require keywords and rules for spam recognition, blacklists have to be populated with IP addresses from known spammers, and Bayesian filters need a training set of spam / ham messages. This input data has to be created when the filter is used first (the "cold-start" problem), and it also has to be adapted continuously to counter the attacks of spammers [113, 215].
2. **Residual error rates,** since current spam filters cannot eliminate the spam problem completely. First, a non-negligible number of spam Emails still reaches the end user, so-called false negatives. Second, some ham messages are discarded because the anti-spam system considers them as spam. Such false positives are especially annoying if the sender of the Email is from the recipient's social community and thus already known to the user, or at least known by somebody else the user knows directly. Therefore, there is a high probability that an Email received from somebody within the social network of the receiver is a ham message. This implies that a social network formed by Email communication can be used as a strong foundation for spam detection.

Even if there existed a perfect anti-spam system, an additional problem would arise for high-volume Email users, some of which simply get too many ham Emails. In these cases, an automated support for Email ranking would be highly desirable. Reputation algorithms are useful in this scenario, because they provide a rating for each Email address, which can subsequently be used to sort incoming Emails. Such ratings can be gained in two ways, globally or personally. The main idea of a global scheme is that people share their personal ratings such that a single global reputation can be inferred for each Email address. The implementation of such a scheme can, for example, be based on network reputation algorithms [112] (see also Section 4.2.2), or on collaborative filtering techniques [180]. In case of a personalized scheme, the output ratings are typically different for each Email user and depend on her personal social network. Such a scheme is reasonable since some people with a presumably high global reputation (e.g., Linus Torvalds) might not be very important in the personal context of a user, compared to other persons (e.g., the project manager).

This section proposes MailRank, a new approach to ranking and classifying Emails exploiting the social network derived from each user's communication circle [31]. We introduce two MailRank variants, both applying a power-iteration algorithm on the Email network graph: Basic MailRank results in a global reputation for each known Email address, and Personalized MailRank computes personalized

values reflecting the point of view of each user. After having discussed the particularities of each approach, the second part of the section analyzes the performance of MailRank under several scenarios, including sparse networks, and shows its resilience against spammer attacks.

4.3.1 The MailRank Algorithm

Bootstrapping the Email Network

As for all reputation algorithms, MailRank needs to start from collecting as many personal votes as possible in order to compute relevant ratings. Generally, this input gathering process should require few or no manual user interactions in order to achieve a high acceptance of the system. Also, the maintenance should require little or no effort at all, thus having the rating of each Email address computed automatically. To achieve these goals, we use already existing data inferred from the communication dynamics, i.e., who has exchanged Emails with whom. We distinguish three information sources as best serving our purposes:

1. **Email Address Books.** If A has the addresses $B_1, B_2, ..., B_n$ in its Address Book, then A can be considered to trust them all, or to vote for them.
2. The **'To:' Fields** of outgoing Emails (i.e., 'To:', 'Cc:' and 'Bcc:'). If A sends Emails to B, then it can be regarded as trusting B, or voting for B. This input data is typically very clean since it is manually selected, while being more accurate than data from address books, which might comprise old or outdated information.
3. **Autowhitelists** created by anti-spam tools (e.g., SpamAssassin) contain a list of all Email addresses from which Emails have been received recently, plus one score for each Email address which determines if mainly spam or ham Emails have been received from the associated Email address. All Email addresses with a high score can be regarded as being trusted.

Figure 4.1 depicts an example Email network graph. Node U_1 represents the Email address of U_1, node U_2 the Email address of U_2, and so on. U_1 has sent Emails to U_2, U_4, and U_3; U_2 has sent Emails to U_1 and U_4, etc. These communication acts are interpreted as trust *votes*, e.g., from U_1 towards U_2, U_4 and U_3, and depicted in the figure using arrows. Building upon such an Email network graph, we can use a power iteration algorithm to compute a *reputation* for each Email address. This can subsequently be used for at least two purposes, namely: (1) Classification into spam and ham Emails, and (2) building a ranking among the remaining ham Emails. Note that it is not necessary for all Email users to participate in MailRank

in order to benefit from it: For example, U_3 does not specify any vote, but still receives a vote from U_1, thus consequently achieving a reputation score.

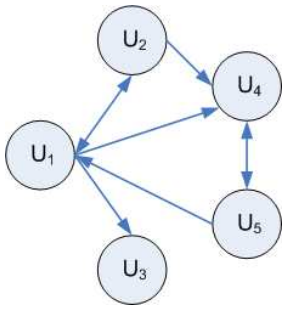

Figure 4.1: Sample Email network

The following subsections provide more information about how the Email reputation scores are generated, first in a global, and then in a personalized perspective. In the end, we briefly sketch the architecture of the system we built to implement MailRank.

Basic MailRank

The main goal of MailRank is to assign a score to each Email address known to the system and to use this score (1) to decide whether each Email is coming from a spammer or not, and (2) to build up a ranking among the filtered non-spam Emails. Its basic version comprises two main steps:

1. Determine a set of Email addresses with a very high reputation in the social network.
2. Run PageRank [172] on the Email network graph, biased on the above set.

It is highly important for the biasing set not to include any spammer. Biasing is a very efficient way to counter malicious collectives trying to attack the system [116, 135]. It can be accomplished in three ways: manually, automatically, or semi-automatically. The manual approach guarantees that no spammers are in the biasing set and provides 100% effectiveness. An automatic selection avoids the costs for the manual selection, but is also error-prone. Finally, a semi-automatic selection starts with the automatic method to generate a biasing set, which is then verified manually to be free of spammers. In our system we take the fully automatic approach, as follows: We first determine the size p of the biasing set by adding the scores of the R nodes with the highest rank such that the resulting sum is equal to 20% of all scores in the system. We additionally limit p to the

minimum between R and 0.25% of the total number of Email addresses[8]. In this way we limit the biasing set to the few most reputable members of the social network, which make the top extremity of the power-law distribution of Email communication links [88, 125].

Algorithm 4.3.1.1. The Basic MailRank Algorithm.

Client Side:
Each vote sent to the MailRank server comprises:
Addr(u) : The hashed version of the Email address of the voter u. Hashing is necessary in order to ensure privacy for the users participating in the social ranking system.
TrustVotes(u) : Hashed version of all Email addresses u votes for (i.e., she has sent an Email to).

Server Side:
1: Combine all received data into a global Email network graph. Let T be the Markov chain transition probability matrix, computed as:
For each known Email address i
If i is a registered address, i.e., user i has submitted her votes
For each trust vote from i to j
$T_{ji} = 1/\text{NumOfVotes}(i)$
Else For each known address j
$T_{ji} = 1/N$, where N is the number of known addresses.
2: Determine the biasing set B (i.e., the most popular Email addr.)
2a: Manual selection or
2b: Automatic selection or
2c: Semi-automatic selection
3: Let $T' = c \cdot T + (1 - c) \cdot E$, with $c = 0.85$ and
$E[i] = [\frac{1}{||B||}]_{N \times 1}$, if $i \in B$, or $E[i] = [0]_{N \times 1}$, otherwise
4: Initialize the vector of scores $\vec{x} = [1/N]_{N \times 1}$, and the error $\delta = \infty$
5: **While** $\delta < \epsilon$, ϵ being the precision threshold
$\vec{x}' = T' \cdot \vec{x}$
$\delta = ||\vec{x}' - \vec{x}||$
6: Output $\vec{x}'$, the global MailRank vector.
7: Classify each Email address in the MailRank network into:
'spammer' / 'non-spammer' based on the threshold T.

---

[8]Both values, the '20%' and the '0.25%' have been determined in extensive tuning simulations.

The final vector of MailRank scores can be used to tag an incoming Email as (1) non-spammer, if the score of the sender address is larger than a threshold T, (2) spammer, if that score is smaller than T, or (3) unknown, if the Email address is not yet known to the system[9]. Each user can adjust T according to her preferred filtering level. If $T = 0$, the algorithm is effectively used to compute the transitive closure of the Email network graph starting from the biasing set. This is sufficient to detect all those spammers for which no user reachable from the biasing set has issued a vote. With $T > 0$, it becomes possible to detect spammers even if some non-spammers vote for spammers (e.g., because the computer of a non-spammer is infected by a virus). However, in this case some non-spammers with a very low rank are at risk of being counted as spammers as well.

The Basic MailRank algorithm is summarized in Algorithm 4.3.1.1.

MailRank with Personalization

As shown in Section 4.3.2, Basic MailRank performs very well in spam detection, while being highly resistant against spammer attacks. However, it still has the limitation of being too general with respect to user ranking. More specifically, it does not address that:

- Users generally communicate with persons ranked average with respect to the overall rankings.
- Users prefer to have their acquaintances ranked higher than other unknown users, even if these latter ones achieve a higher overall reputation from the network.
- There should be a clear difference between a user's communication partners.

Personalizing on each user's acquaintances tackles these aspects. Its main effect is boosting the weight of user's votes, while decreasing this influence for all the other votes. Thus, the direct communication partners will achieve much higher ranks, even though initially they were not among the highest ones. Moreover, due to the rank propagation, their votes will have a high influence as well.

Now that we have captured the user requirements mentioned, we should also focus our attention on a final design issue of our system: scalability. Simply biasing MailRank on user's acquaintances will not scale well, because it must be computed for each preference set, that is for every registered user. Jeh and Widom [131] have

[9]To allow new, unknown users to participate in MailRank, an automatically generated Email could be sent to the unknown user encouraging her to join MailRank (challenge-response scheme), thus bringing her into the non-spammer area of reputation scores.

proposed an approach to calculate Personalized PageRank vectors, which can also be adapted to our scenario, and which can be used with millions of subscribers. To achieve scalability, the resulting personalized vectors are divided in two parts: one common to all users, precomputed and stored off-line (called "partial vectors"), and one which captures the specifics of each preference set, generated at run-time (called "hubs skeleton"). We will have to define a restricted set of users on which rankings can be biased though (we shall call this set "hub set", and note it with H)[10]. There is one partial vector and one hub skeleton for each user from H. Once an additional regular user registers, her personalized ranking vector will be generated by reading the already precomputed partial vectors corresponding to her preference set (step 1), by calculating their hubs skeleton (step 2), and finally by tying these two parts together (step 3). Both the algorithm from step 1 (called "Selective Expansion") and the one from step 2 (named "Repeated Squaring") can be mathematically reduced to biased PageRank. The latter decreases the computation error much faster along the iterations and is thus more efficient, but works only with the output of the former one as input. In the final phase, the two sub-vectors resulted from the previous steps are combined into a global one. The algorithm is depicted in the following lines. To make it clearer, we have also collected the most important definitions it relies on in Table 4.1.

Term	Description
Set V	The set of all users.
Hub Set H	A subset of users.
Preference Set P	Set of users on which to personalize.
Preference Vector p	Preference set with weights.
Personalized PageRank Vector (PPV)	Importance distribution induced by a preference vector.
Basis Vector r_u	PPV for a preference vector with a single nonzero entry at u.
Hub Vector r_u	Basis vector for a hub user $u \in H$.
Partial Vector $r_u - r_u^H$	Used with the hubs skeleton to construct a hub vector.
Hubs Skeleton $r_u(H)$	Used with partial vectors to construct a hub vector.

Table 4.1: Terms specific to Personalized MailRank.

[10]Note that an improved version of this algorithm has been proposed recently by Sarlos et al. [190], thus eliminating the limitation on the size of the biasing set.

Algorithm 4.3.2.2. Personalized MailRank.

0: (Initializations) Let u be a user from H, for which we compute the partial vector and the hubs skeleton. Also, let $D[u]$ be the approximation of the basis vector corresponding to user u, and $E[u]$ the error of its computation.
Initialize $D_0[u]$ with:

$$D_0[u](q) = \begin{cases} c = 0.15 & , q \in H \\ 0 & , otherwise \end{cases}$$

Initialize $E_0[u]$ with:

$$E_0[u](q) = \begin{cases} 1 & , q \in H \\ 0 & , otherwise \end{cases}$$

1: (Selective Expansion) Compute the partial vectors using
$Q_0(u) = V$ and $Q_k(u) = V \setminus H$, for $k > 0$, in the formulas below:
$\mathbf{D_{k+1}[u]} = \mathbf{D_k[u]} + \sum_{q \in Q_k(u)} c \cdot E_k[u](q) \cdot \mathbf{x_q}$
$\mathbf{E_{k+1}[u]} = \mathbf{E_k[u]} - \sum_{q \in Q_k(u)} E_k[u](q)\mathbf{x_q} + \sum_{q \in Q_k(u)} \frac{1-c}{|O(q)|} \sum_{i=1}^{|O(q)|} E_k[u](q) \cdot \mathbf{x}_{O_i(q)}$
Under this choice, $D_k[u] + c \cdot E_k[u]$ will converge to $\mathbf{r_u} - \mathbf{r_u^H}$,
the partial vector corresponding to u.

2: (Repeated squaring) Having the results from the first step as input, one can now compute the hubs skeleton ($r_u(H)$). This is represented by the final $D[u]$ vectors, calculated using $Q_k(u) = H$ into:
$\mathbf{D_{2k}[u]} = \mathbf{D_k[u]} + \sum_{q \in Q_k(u)} E_k[u](q) \cdot D_k[q]$
$\mathbf{E_{2k}[u]} = \mathbf{E_k[u]} - \sum_{q \in Q_k(u)} E_k[u](q) \cdot \mathbf{x_q} + \sum_{q \in Q_k(u)} E_k[u](q) \cdot E_k[q]$
As this step refers to hub-users only, the computation of $\mathbf{D_{2k}[u]}$ and $\mathbf{E_{2k}[u]}$ should consider *only* the components regarding users from H, as it significantly decreases the computation time.

3: Let $p = \alpha_1 u_1 + \ldots + \alpha_z u_z$ be a preferenced vector, where u_i are from H and i is between 1 and z, and let:

$$r_p(h) = \sum_{i=1}^{z} \alpha_i (r_{u_i}(h) - c \cdot x_{p_i}(h)),\ h \in H$$

which can be computed from the hubs skeleton.
The PPV v for p can then be constructed as:

$$v = \sum_{i=1}^{z} \alpha_i (r_{u_i} - r_{u_i}^H) + \frac{1}{c} \sum_{h \in H\ r_p(h) > 0} r_p(h) \cdot \left[(\mathbf{r_u} - \mathbf{r_u^H}) - c \cdot x_h \right]$$

MailRank System Architecture

MailRank is composed of a server, which collects all user votes and delivers a score for any known Email address, and an Email proxy on the client side, which interacts with the MailRank server.

The MailRank Server collects the input data (i.e., the votes) from all users to run the MailRank algorithm. Votes are assigned with a lifetime for (1) Identifying and deleting Email addresses which have not been used for a long time, and (2) Detecting spammers which behave good for some time to get a high rank and start to send spam Emails afterwards.

The MailRank Proxy resides between user's Email client and her regular local Email server. It performs two tasks: When receiving an outgoing Email, it first extracts the user's votes from the available input data (e.g., by listening to ongoing Email activities or by analyzing existing sent-mail folders). Then, it sends the votes to the MailRank server and forwards the Email to the local Email server. To increase efficiency, only those votes that have not been submitted yet (or that would expire otherwise) are sent. Also, for privacy reasons, votes are encoded using hashed versions of Email addresses. Upon receiving an Email, the proxy queries the MailRank server about the ranking of the sender address (if not cached locally) and classifies / ranks the Email accordingly.

Note that one could also make use of secure signing schemes to enable analyzing both outgoing and incoming Emails for extracting "votes"[11]. This helps not only to bootstrap the system initially, but also introduces the votes of spammers into MailRank. Such votes have a very positive aspect, since they increase the score for the spam recipients (i.e., non-spammers). Thus, spammers face more difficulties to attack the system in order to increase their own rank.

4.3.2 Experiments

Experimental Setup

Real-world data about Email networks is almost unavailable because of privacy reasons. Yet some small studies do exist, using data gathered from the log files of a student Email server [88], or of a comany wide server [125], etc. In all cases, the analyzed Email network graph exhibits a power-law distribution of in-going (exponent 1.49) and out-going (exponent 1.81) links.

[11] Analyzing incoming votes raises more security issues since we need to ensure that the sender did indeed vote for the recipient, i.e., the Email is not faked. This can be achieved by extending current sender authentication solutions.

To be able to vary certain parameters such as the number of spammers, we evaluated MailRank[12] using an extensive set of simulations, based on a power-law model of an Email network, following the characteristics presented in the above mentioned literature studies. Additionally, we used an exponential cut-off at both tails to ensure that a node has at least five and at most 1500 links to other nodes, which reflects the nature of true social contacts [125]. If not noted otherwise, the graph consisted of 100,000 non-spammers[13] and the threshold T was set to 0. In a scenario without virus infections, this is sufficient to detect spammers and to ensure that non-spammers are not falsely classified. Furthermore, we repeated all simulations for at least three times with different randomly generated Email networks to determine average values. Our experiments focused on three aspects: Effectiveness in case of very sparse MailRank networks (i.e., only few nodes submit votes, the others only receive votes), exploitation of spam characteristics, and attacks on MailRank.

Very Sparse MailRank Networks

In sparse MailRank networks, a certain amount of Email addresses only receive votes, but do not provide any because their owners do not participate in MailRank. In this case, some non-spammers in the graph could be regarded as spammers, since they achieve a very low score.

To simulate sparse MailRank networks, we created a full graph as described above and subsequently deleted the votes of a certain set of Email addresses. We used several removal models:

- All: Votes can be deleted from all nodes.
- Bottom99.9%: Nodes from the top 0.1% are protected from vote deletion.
- Avg: Nodes having more than the average number of outgoing links are protected from vote deletion.

The first model is rather theoretical, as we expect the highly-connected non-spammers to register with the system first[14]. Therefore, we protected the votes of the top nodes in the other two methods from being deleted[15]. Figure 4.2 depicts

[12]As personalization brings a significant improvement only in creating user-specific rankings of Email addresses (i.e., it produces only minimal improvements for spam detection), we used only Basic MailRank within the analysis.

[13]We also simulated using 10,000 and 1,000,000 non-spammers and obtained very similar results.

[14]Such behavior was also observed in real-life systems, e.g., in the Gnutella P2P network (http://www.gnutella.com/).

[15]The 100% from 'Bottom99.9%' and 'avg' actually refer to 100% of the non-protected nodes.

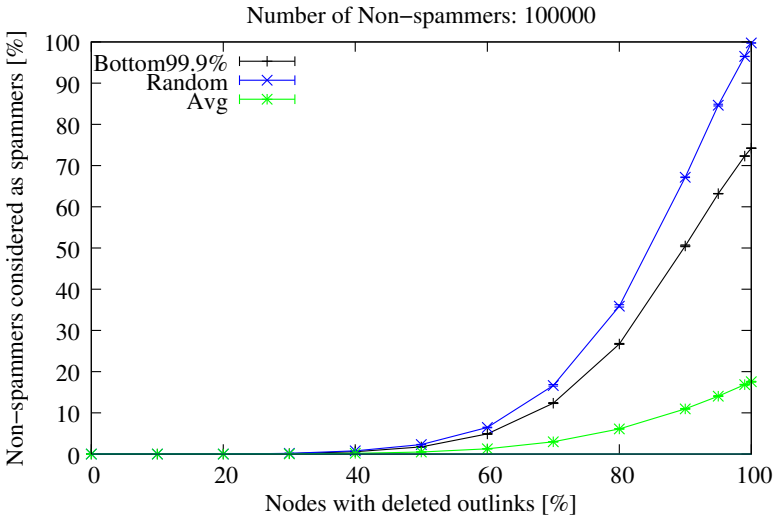

Figure 4.2: Very sparse MailRank networks.

the average percentage of non-spammers regarded as spammers, depending on the percentage of nodes with deleted votes. Non-spammers registered to the system will be classified as spammers only when very few, non-reputable MailRank users send them Emails. As studies have shown that people usually exchange Emails with at least five partners, such a scenario is rather theoretical. However, as the power-law distribution of Email communication is expected only after the system has run for a while, we intentionally allowed such temporary anomalies in the graph. Even though for high deletion rates ($70-90\%$) they resulted in some non-spammers being classified as spammers, MailRank still performed well, especially in the more realistic 'avg' scenario (the bigger error observed in the theoretical 'Random' scenario was expected, since random removal may result in the deletion of high-rank nodes contributing many links to the social network).

Exploitation of Spam Characteristics

If we monitor current spammer activities (i.e., sending Emails to non-spammers), the Emails from spammers towards non-spammers can be introduced into the system as well. This way, spammers actually contribute to improve the spam detection capabilities of MailRank: The more new spammer Email addresses and Emails are introduced into the MailRank network, the higher they increase the score of the receiving non-spammers. This can be seen in a set of simulations with 20,000 non-spammer addresses and a varying number of spammers (up to 100,000, as depicted in Figure 4.3), where the rank of the top 0.25% non-spammers

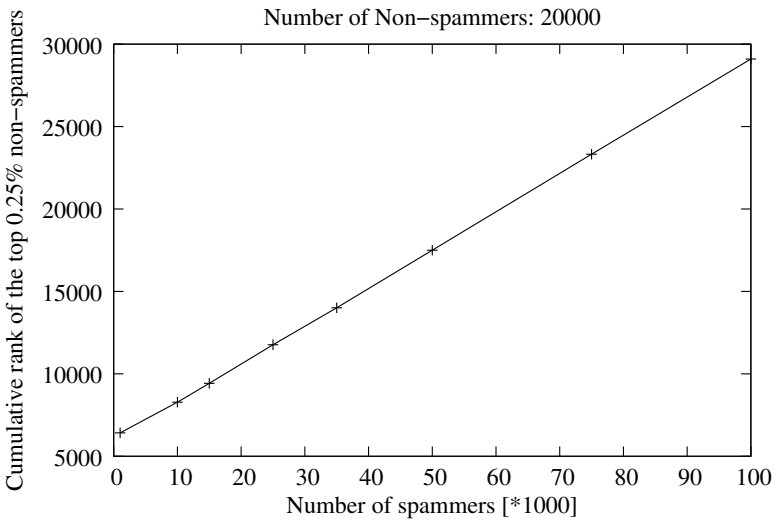

Figure 4.3: Rank increase of non-spammer addresses.

increases linearly with the number of spammer addresses included in the MailRank graph.

Attacking MailRank

In order to be able to attack MailRank, spammers must receive votes from other MailRank users to increase their rank. As long as nobody votes for spammers, they will achieve a minimal score and will thus be easily detected. This leaves only two ways of attacks: formation of malicious collectives and virus infections.

Malicious collectives. The goal of a malicious collective is to aggregate enough score into one node to push it into the biasing set. If no manually selected biasing set can be used to prevent this, one of the already many techniques to identify Web link farms could be employed (see for example [216, 44]). Furthermore, we require MailRank users willing to submit their votes to manually register their Email address(es). This impedes spammers to automatically register millions of Email addresses in MailRank and also increases the cost of forming a malicious collective. To actually determine the cost of such a manual registration, we have simulated a set of malicious users as shown in Figure 4.4. The resulting position of node 1, the node that should be pushed into the biasing set, is depicted in Figure 4.5 for an Email network of 20,000 non-spammers, malicious collectives of 1000 nodes each, and an increasing number of collectives on the x-axis. When there are few large-scale spammer collectives, the system could be relatively easy attacked.

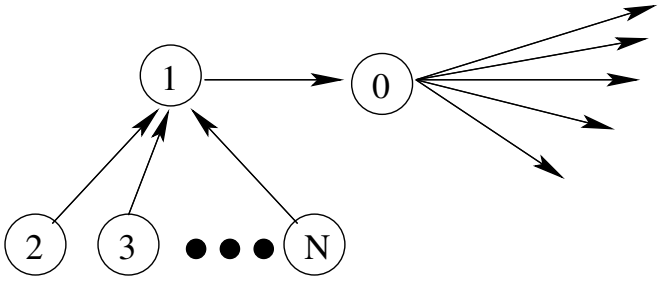

Figure 4.4: Malicious collective: nodes 2–N vote for node 1 to increase the rank of node 1 and node 1 itself votes for node 0, the Email address that is finally used for sending spam Emails.

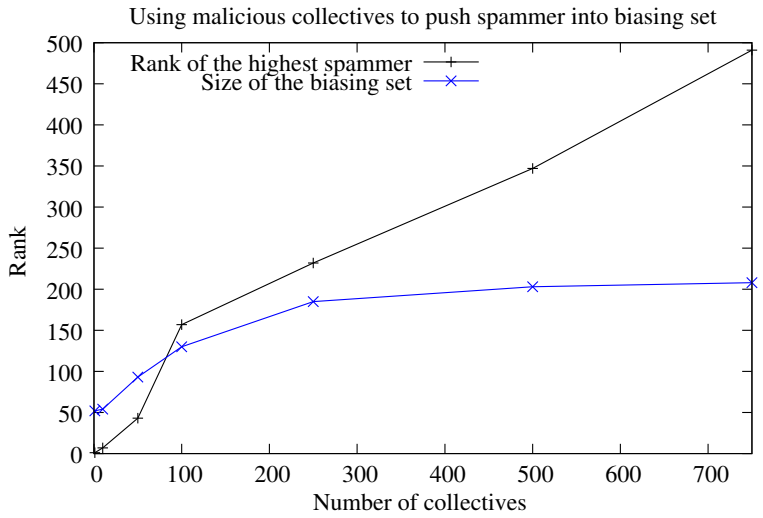

Figure 4.5: Automatic creation of the biasing set.

However, as users must manually register to the system, forming a collective of sufficient size is practically infeasible. Moreover, in a real scenario there will be more than one malicious collective, in which case pushing a node into the biasing set is almost impossible: As shown in Figure 4.5, it becomes more difficult for a malicious collective to push one node into the biasing set, the more collectives exist in the network. This is because the spammers registered to the system implicitly vote for the non-spammers upon sending them (spam) Emails. This way, the rank of the best spammer increases, i.e., it achieves a lower reputation throughout the network, and thus it has lower chances of being accepted into the biasing set.

Virus infections. Another possible attack is to make non-spammers vote for spammers. To counter incidental votes for spammers (e.g., because of a misconfigured vacation daemon), an additional confirmation process could be required if

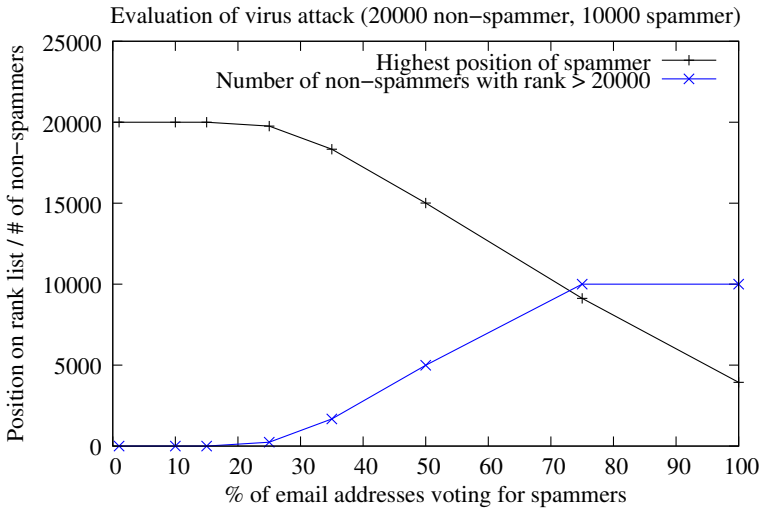

Figure 4.6: Simulation results: Virus attack.

a vote for one particular Email address would move that address from 'spammer' to 'non-spammer'. However, spammers could still *pay* non-spammers to send spam on their behalf. Such an attack can be successful initially, but the rank of the non-spammer addresses will decrease after some time to those of spammers, due to the limited life time of votes. Finally, one could use virus / worm technology to infect non-spammers and make them vote for spammers. We simulated such an attack according to Newman's studies [168], which showed that when the 10% most connected members of a social network are not immunized (e.g., using anti-virus applications) worms would spread too fast. The results are shown in Figure 4.6 with a varying amount of non-spammers voting for 50% of all spammers. If up to about 25% of the non-spammers are infected and vote for spammers, there is still a significant difference between the ranks of non-spammers and spammers, and no spammer manages to get a higher rank than the non-spammers. If more than 25% non-spammers are infected, the spammer with the highest rank starts to move up in the rank list (the upper line from Figure 4.6 descends towards rank 1). Along with this, there will be no clear separation between spammers and non-spammers, and two threshold values must be employed: one MailRank score T_1 above which all users are considered non-spammers and another one $T_2 < T_1$ beneath which all are considered spammers, the members having a score within (T_1, T_2) being classified as unknown.

4.3.3 Discussion

Email spam detection remains a serious problem for PC users. Many approaches have been developed, yet each time spammers came up with new methods to bypass Email filtering. In this section we proposed MailRank, a new Email ranking and classification scheme, which intelligently exploits the social communication network created via Email interactions. On the resulting Email network graph, input values are collected from the sent-mail folder of all participants, as well as from other sources, and a power-iteration algorithm is used to rank trustworthy senders and to detect spammers. MailRank brings the following advantages upon previous spam filtering techniques:

- **Shorter individual cold-start phase**. If a MailRank user does not know an Email address X, MailRank can provide a rank for X as long as at least another MailRank user has provided information about it. Thus, the so-called "cold-start" phase, i.e., the time a system has to learn until it becomes functional, is reduced: While most successful anti-spam approaches (e.g., Bayesian filters) have to be trained for each single user (in case of an individual filter) or a group of users (for example, in case of a company-wide filter), MailRank requires only a single global cold start phase when the system is bootstrapped. In this sense it is similar to globally managed whitelists, but it requires less administrative efforts to manage the list and it can additionally provide information about *how good* an Email address is, and not only a classification into "good" or "bad".
- **High attack resilience**. MailRank is based on a power iteration algorithm, which is typically resistant against attacks.
- **Partial participation**. Building on the power-law nature of Email networks, MailRank can compute a rank for a high number of Email addresses even if only a subset of Email users actively participates in the system.
- **Stable results**. Social networks are typically rather stable, so the computed ratings of the Email addresses will usually also change slowly over time. Hence, spammers need to behave well for quite some time to achieve a high rank. Though this cannot resolve the spam problem entirely (in the worst case, a spammer could, for example, buy Email addresses from people who have behaved well for some time), it will increase the cost for using new Email addresses.
- **Can reduce load on Email servers**. Email servers do not have to process the Email body to detect spam. This significantly reduces the computational power for spam detection compared to, for example, content-based approaches or collaborative filters [146].

- **Personalization**. In contrast to spam classification approaches that distinguish only between 'spam' and 'non-spam', ranking also enables personalization features. This is important since there are certain Email addresses (e.g., newsletters), which some people consider to be spammers while others do not. To deal with such cases, a MailRank user can herself decide about the score threshold below which all Email addresses are considered spammers. Moreover, she could use two thresholds to determine spammers, non-spammers, and unclear classifications. Furthermore, she might want to give more importance to her relatives or to her manager, than to other unrelated persons with a globally high reputation.
- **Scalable computation**. Power iteration algorithms have been shown to be computationally feasible even when personalized over very large graphs [190].
- **Can also counter other forms of spam**. When receiving spam phone calls (SPIT[16]), for example, it is impossible to analyze the content of the call before accepting / rejecting it. At best only the caller identifier is available, which is similar to the sender Email address. MailRank can be used to analyze the caller ID and decide whether a caller is a spammer or not.

Our experiments showed MailRank to perform well in the presence of very sparse networks: Even in case of a low participation rate, it can effectively distinguish between spammer Email addresses and non-spammer ones, even for those users not participating actively in the system. MailRank proved itself to be also very resistant against spammer attacks and, in fact, has the property that when more spammer Email addresses are introduced into the system, the spam detection performance increases.

In the future, one could move the implementation from a centralized system to a distributed one in order to allow for more scalability and to avoid bottlenecks. From the algorithmic perspective, some of the already existing Web anti-spam approaches could be built on top of MailRank, so as to ensure an increased attack resistance of the system.

4.4 Ranking for Web Spam Detection

Although Email spam is already widely present in our lives, another "industry" is emerging at an even faster pace: (Web) Search Engine Optimization, shortly SEO [115]. Given the increasing importance of search engines in modern society, many online organizations currently attempt to artificially increase their rank, since

[16]Spam over Internet Telephony, http://www.infoworld.com/article/04/09/07/HNspamspit_1.html

a higher rank implies more users visiting their Web pages, which subsequently implies an increased profit. This results in a strong negative impact upon the output of our everyday Web searches, making the high quality pages harder to find and the low quality ones more accessible.

Search engines adopt several different sources of evidence to rank the Web pages matching a user query, such as textual content, title of Web pages, anchor text information, or the link structure of the Web [28]. Each of them is generally attacked differently by spammers. As in the entire thesis, in this section we focus again on link analysis, and thus tackle the latter measure, which is in fact one of the most useful sources of evidence adopted. To extract information from the link structure, search engines use algorithms that assess the quality (or popularity) of Web pages by analyzing the linkage relationships among them. The success of this strategy relies on the assumption that a link to a Web page represents a vote from a user that sustains the quality of that targeted page.

In spite of the success of link analysis algorithms, many artificial hyperlink structures lead these algorithms to provide wrong conclusions about the quality of Web pages. This phenomenon happens because links that cannot be interpreted as votes for quality sometimes negatively affect the search engine ranking results. Such links are called nepotistic links (or spam links), i.e., links *intentionally* created to artificially boost the rank of some given set of pages, usually referred to as *spam pages* [216].

In this section we propose a site-level approach for detecting generic spam links on the Web. Previous algorithms have focused on identifying spam only by analyzing page level relationships, which clearly misses some of the higher level information, generated between a group of sites. We investigate three main types of site level relationships: mutual reinforcement (in which many links are exchanged between two sites), abnormal support (where most of one site's links are pointing to the same target site), and link alliances (in which several sites create complex link structures that boost the PageRank score of their pages). When the relation between such sets of sites is considered suspicious, we assume that the links between them are nepotistic and penalize them accordingly. Finally, it is important to note that this new approach is complementary to the existing page level approaches, and both strategies should be adopted simultaneously for identifying spam links in a search engine database.

We will now proceed with presenting our three approaches to site level spam detection. Then, we will continue with an extensive evaluation of these algorithms, followed by a discussion about their strengths and weaknesses, as well as about possible extesions which could be built on top of them.

4.4.1 Site Level Spam Detection

We argue here that many spam links can be easily detected when the relationships between sites, instead of pages, are analyzed. Even though the current page centered approaches for detecting spam still hold (and will also be needed in the future), they may not be the best solution to deal with many practical situations. For example, a company having two branches with different sites could easily establish many links between its two sites in order to boost their PageRank. These would be regarded as true votes by the current ranking approaches, even though they connect two entities having the same owner. Even when this would occur accidentally (in which case we are dealing with "noisy" links instead; see also Section 4.2.3 for more details on noisy links), such relationships are still artificial and should not be included in the search engine ranking computation. Worse, automatically generated complex site level link spam structures may be missed by the current page level approaches. Therefore, we propose detecting spam at a *site level* rather than at a page level, investigating the above mentioned three types of artificial site level relationships. The following sections detail a separate analysis on each of these constructs.

Site Level Mutual Reinforcement

Our first site level approach to detect spam links on Web collections is based on the study of how connected are pairs of sites. Our assumption in this approach is that when two sites are strongly connected, they artificially boost their results in link analysis algorithms. We name this phenomenon as a *site level mutual reinforcement*. Mutual reinforcement relations have been tackled as early as 1998 by Bharat and Henzinger [27]. However, all approaches proposed so far are centered around the Web page as a unit item. We therefore study the mutual reinforcement problem at the site level, because a considerable amount of spam links between these type of Web sites cannot be detected using approaches working at the page level. We thus consider all links between strongly connected sites as spam, including links between individual pages that are not suspicious per se. This is because these links artificially boost the popularity rank of the pages belonging to the pair of suspicious Web sites. Let us now discuss the two different algorithms we propose for detecting mutual site reinforcement relationships.

Bi-Directional Mutual Site Reinforcement (BMSR). This algorithm takes into account the number of link exchanges between pages from the two studied sites. We say that two pages p_1 and p_2 have a link exchange if there is a link from p_1 to p_2 and a link from p_2 to p_1. Our first method tries to identify site pairs that have an abnormal amount of link exchanges between their pages. In

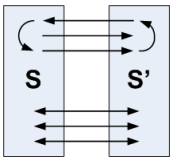

Figure 4.7: Example of site level link exchanges.

these cases, we consider the pair as suspicious and all the links between its sites are considered spam. The threshold to consider a pair of sites suspicious is set through experiments.

Uni-Directional Mutual Site Reinforcement (UMSR). As sites are large structures, we also investigate the possibility of relaxing the notion of "link exchange" into "link density", i.e., counting all links between two sites, disregarding their orientation. This ensures capturing sites attempting to boost the ranks of their pages without necessarily constructing link exchanges. One of the many possibilities to achieve this goal is depicted on the upper side of Figure 4.7, using cycles. The connection between two sites may create a set of cycles on the Web graph containing pages from both of them. It is known that such cycle structures boost the popularity of Web pages [114], and since many cycles can arise from strongly connected sites, such alliances between sites create anomalies in the final PageRank.

And yet this measure might be too drastic! For instance, many pages of a site might have a link to Yahoo! Search just because they think this is a good service. Since all links between the two sites are counted, it does not matter if Yahoo! does not link back to the above mentioned site. We therefore propose a more comprehensive measure, which returns the minimum between the amount of links from a site s to some site s', and the amount of links coming back from s' to s. We call this "mutual link density".

On the example from Figure 4.7, there are 3 link exchanges between sites s and s' and the link density is 9 (link exchanges are also counted). In order to calculate these values, one needs to iterate over all pages, and for each page to increment the site level statistics every time a link exchange is found (see Algorithm 4.4.2.1 below, lines 5-8), for BMSR, or simply every time a link is encountered (Algorithm 4.4.2.1, lines 5-6, and 9), for UMSR. Note that Algorithm 4.4.2.1 computes the link density as a measure of UMSR. In order to obtain the mutual link density, one would have to calculate $UMSR(s, s')$ and $UMSR(s', s)$ separately, and then return their minimum as a result.

Algorithm 4.4.2.1. Detecting Link Exchanges at Site Level.

1: Let $BMSR(s, s')$ and $UMSR(s, s')$ denote the amount of link exchanges and the link density between sites s and s' respectively.
2: **For** each site s
3: **For** each site $s' \neq s$
4: $BMSR(s, s') = UMSR(s, s') = 0$
5: **For** each page $p \in V$, p residing on site s
6: **For** each page $q \in Out(p)$, q from site $s' \neq s$
7: **If** $p \in Out(q)$
8: **Then** $BMSR(s, s') = BMSR(s, s') + 1$
9: $UMSR(s, s') = UMSR(s', s) = UMSR(s, s') + 1$

Computing Page Ranks. Let us now see how we could use these measures to improve PageRank quality. An approach is depicted in Algorithm 4.4.2.2, which removes all links between all pairs of sites (s, s'), if the BMSR or UMSR values between them are above a certain threshold. In our experiments, we used 10, 20, 50, 100, 250 and 300 for link density (250 being best, yet still with poor performance), and 2, 3 and 4 for link exchanges (with 2 having better results, indicating that most sites exchange incorrect votes, or links, with only a few partners, like a company with its branches).

Algorithm 4.4.2.2. Removing Site-Level Link Exchanges.

1: **For** each site s
2: **For** each site s'
3: **If** $*MSR(s, s') \geq \epsilon_{*MSR*}$
4: **Then** Remove all links between s and s'
5: Compute regular PageRank.

Site Level Abnormal Support

Another type of situation we consider is the *site level abnormal support*(**SLAbS**). It occurs when a single site is responsible for a high percentage of the total amount of links pointing to another site. This situation can easily arise within a Web collection. For instance, and unfortunately, once the spammers have read the previous section, they could start to seek for new schemes that circumvent

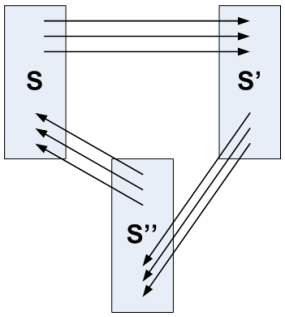

Figure 4.8: Example of site chains.

the algorithms we presented. A relatively simple approach they could take is to create chains of sites supporting each other through a limited number of links (see Figure 4.8 for an example). This is because their space of available choices is diminishing: Using too many links would make them detectable by our site level mutual reinforcement algorithms above, while using other structures than chains (e.g., hierarchy of sites) would visibly make their success more costly. We therefore propose the following axiom:

Axiom 1 *The total amount of links to a site (i.e., the sum of links to its pages) should not be strongly influenced by the links it receives from some other site.*

In other words, for any site s there should not be a site $s' \neq s$, whose number of links towards s is above a certain percentage of the total number of links s receives overall. In our experiments we tested with thresholds ranging from 0.5% up to 20% of the total number of links to s and the best results were achieved at 2%. Whenever such a pair of sites (s, s') is found, all links between them are marked as spam. Note that links from s to s' are also taken as spam because we consider the relation between them suspicious. After this trimming process is over, we remove the detected spam links and the regular PageRank is run over the cleaned link database. The approach is summarized in Algorithm 4.4.2.3.

Algorithm 4.4.2.3. Removing Site-Level Abnormal Support(SLAbS).

1: **For** each site s
2: **Let** t be the total number of links to pages of s
3: **For** each site s' that links to s
4: **Let** $t_{(s',s)}$ be the number of links from s' to s, and **Let** $supp = t_{(s',s)}/t$
5: **If** $supp \geq \epsilon_{AS}$
6: **Then** Remove all links between s' and s
7: Compute regular PageRank.

Site Level Link Alliances

Another hypothesis we considered is that the popularity of a site cannot be supported only by a group of strongly connected sites. The intuition behind this idea is that a Web site is as popular as diverse and independent are the sites that link to it. In fact, as we will see from the experiments section, our algorithm which detects and considers this concept of independence when computing PageRank gives a strong improvement in the overall quality of the final rankings.

Further, continuing the scenario discussed in the previous Section, suppose spammers do have enough resources available to build complex hierarchies of sites that support an end target site, as illustrated in Figure 4.9. These hierarchies have previously been named *Link Spam Alliances* by Gyögyi and Garcia-Molina [114], but they did not present any solution to counteract them. Such structures would generate sites linked by a strongly connected community, thus contradicting our general hypothesis about the relation between diversity of sites that link to a site and its actual popularity.

Before discussing our approach, we should note that we do not address page level link alliances, i.e., hierarchies of pages meant to support an end target page, all pages residing on the same site, or on very few sites. These types of structures could be easily annihilated for example by using different weights for intra-site and inter-site links, or by implementing the approach presented by Bharat and Henzinger in [27], where every in-link of some page p is assigned the weight $1/k$ if there are k pages pointing to p (for link alliances distributed over several sites).

The more complicated situation is to find link alliances (intentional or not) over several sites, as the one depicted in Figure 4.9 (boxes represent sites). Our intuition is that these alliances would still have to consist of highly interconnected pages. More specifically, if a page p has in-links from pages i_1, i_2, ..., i_I, and these latter pages are highly connected, then they are suspect of being part of

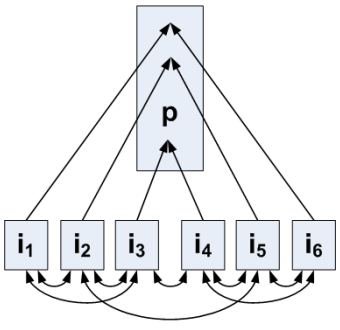

Figure 4.9: Example of link alliance spanning over several sites.

a structure which could deceive popularity ranking algorithms. We evaluate the degree of susceptivity using the following algorithm:

Algorithm 4.4.2.4. Computing Site-Level Link Alliance Susceptivity.

1: **For** each page p
2: Let Tot count the number of out-links of all pages $q \in In(p)$
3: Let $TotIn$ count the number of out-links of all pages $q \in In(p)$, such that they point to some other page from $In(p)$
4: **For** each page $q \in In(p)$
5: **For** each page $t \in Out(q)$
6: $Tot = Tot + 1$
7: **If** $t \in In(p)$
8: **Then** $TotIn = TotIn + 1$
9: Susceptivity(p) = $\frac{TotIn}{Tot}$.

Once the susceptivity levels are computed, we downgrade the in-links of every page p with $(1 - Susceptivity(p))$, uniformly distributing the remaining votes to all pages. This latter step is necessary in order to ensure the convergence of the Markov chain associated to the Web graph, i.e., to ensure the sum of transition probabilities from each state st remains 1. The entire approach is also presented in Algorithm 4.4.2.5.

Algorithm 4.4.2.5. Penalizing Site-Level Link Alliances.

1: Let $PR(i) = 1/\|V\|, \forall i \in \{1, 2, ..., \|V\|\}$
2: **Repeat** until convergence
3: **For** each page p
4: $PR(p) = (1 - Susceptivity(p)) \cdot c \cdot \sum_{q \in In(p)} \frac{PR(q)}{\|Out(q)\|} + \frac{1-c}{\|V\|}$
5: $Residual = Susceptivity(p) \cdot c \cdot \sum_{q \in In(p)} \frac{PR(q)}{\|Out(q)\|}$
6: **For** each page p'
7: $PR(p') = PR(p') + \frac{Residual}{\|V\|}$

4.4.2 Experiments

Experimental Setup

We evaluated our Web spam detection techniques on the link database of the TodoBR search engine[17] (now Google Brazil). This database consisted of a collection of 12,020,513 pages extracted from the Brazilian Web, connected by 139,402,245 links. As it represents a considerably connected snapshot of the Brazilian Web community, which is probably as diverse in content and link structure as the entire Web, we think it makes a realistic testbed for our experiments.

In order to evaluate the impact of our algorithms within practical situations, we extracted test queries from the TodoBR log, which is composed of 11,246,351 queries previously submitted to the search engine. We divided these selected queries in two groups:

1. *Bookmark queries*, in which a specific Web page is sought;
2. *Topic queries*, in which people are looking for information on a given topic, instead of some page.

Each query set was further divided in two subsets, as follows:

- *Popular queries*: Here, we selected the top most popular bookmark / topic queries found in the TodoBR log. These queries usually search for well known Web sites and are useful to check what happens to these most commonly searched pages after the spam detection methods have been applied.
- *Randomly selected queries*: In this scenario we selected the queries randomly. These queries tend to search for less popular sites and show the impact of our techniques on pages that are probably not highly ranked by PageRank.

Then, 14 undergraduate and graduate computer science students (within different areas) evaluated the selected queries under various experimental settings. All of them were familiar with the Brazilian Web pages and sites, in order to ensure more reliability to our experiments.

The bookmark query sets contained each 50 queries, extracted using the above mentioned techniques. All bookmark query results were evaluated using MRR (Mean Reciprocal Ranking), which is the metric adopted for bookmark queries on the TREC Conference[18] and is computed by the following equation:

$$MRR(QS) = \frac{\sum_{\forall q_i \in QS} \frac{1}{PosRelAns(q_i)}}{|QS|} \tag{4.2}$$

[17] http://www.todobr.com.br/
[18] http://trec.nist.gov/

where QS is the set of queries we experiment on, and $PosRelAns(q_i)$ is the position of the first relevant answer in the rankings output for query q_i. MRR is the most common metric for evaluating the quality of results in bookmark queries. As it can be seen, its formula prioritizes methods that obtain results closer to the top of the ranking, adopting an exponential reduction in the scores (i.e., higher scores are better), as the position of the first relevant answer in the ranking increases. Also, MRR is very good at assessing the "real life" performance of the search engine, since the most important URLs are those placed at the top of the search output. However, MRR is not sensible to pages having huge drops in position (e.g., from place 15 to place 40). Therefore, we also adopted another measure, mean position, (denoted MEANPOS in the tables to follow), which computes the average position of the first relevant answer in the output provided for each query. This metric results in a linear increase in the scores (higher is worse) as the position of the relevant answer increases.

For topic queries, we used two sets of 30 queries also selected from the TodoBR log as described previously. These different queries evaluate the impact of our spam detection algorithms when searching for some given topics. In this case, we evaluated the results using the same pooling method as within the Web Collection of TREC [121]. We thus constructed query pools containing the first top 20 answers for each query and algorithm. Then, we assessed our output in terms of various precision based metrics: For each algorithm, we evaluated the Mean Average Precision (MAP), the precision at the first 5 positions of the resulted ranking (P@5), as well as the precision at the top 10 output rankings (P@10). In all cases the relevant results were divided in two categories, (1) relevant and (2) highly relevant. Also, we processed all queries according to the user specifications, as extracted from the TodoBR log: phrases, Boolean conjunctive or Boolean disjunctive. The set of documents achieved for each query was then ranked according to the PageRank algorithm, with and without each of our link removal techniques applied. Finally, all our results were tested for statistical significance using T-tests (i.e., we tested whether the improvement over PageRank without any links removed is statistically significant).

In all forthcoming tables, we will label the algorithms we evaluated as follows:

- **ALL LINKS**: No spam detection.
- **UMSR**: Uni-directional Mutual Site Reinforcement.
- **BMSR**: Bi-directional Mutual Site Reinforcement.
- **SLAbS**: Site Level Abnormal Support.
- **SLLA**: Site Level Link Alliances.
- Combinations of the above, in which every method is applied independently to remove (UMSR, BMSR, SLAbS) or downgrade (SLLA) links.

Method	Threshold
UMSR	250
BMSR	2
SLAbS	2%

Table 4.2: Best thresholds found for each algorithm using MRR as the tuning criterion.

Algorithm specific aspects. Another important setup detail is to divide the collection in Web sites, as the concept of *Web site* is rather imprecise. In our implementation, we adopted the host name part of the URLs as the keys for identifying individual Web sites. This is a simple, yet very effective heuristic to identify sites, as pages with different host names usually belong to different sites, while those with identical host names usually belong to the same site.

As UMSR, BMSR and SLAbS all use thresholds to determine whether links between pairs of sites are spam or not, it is important to tune such thresholds in order to adjust the algorithms to the collection in which they are applied. For the experiments we performed, we adopted the MRR results achieved for bookmark queries as the main parameter to select the best threshold. This metric was adopted because the link information tends to have a greater impact on bookmark queries than on topic queries. Further, MRR can be calculated automatically, reducing the cost for tuning. The best parameters for each method depend on the database, the amount of spam and the requirements of the search engine where they will be applied.

Table 4.2 presents the best thresholds we found for each algorithm using the MRR as the tuning criteria. These parameters were adopted in all the experiments presented.

Results

Bookmark Queries. We evaluated the bookmark queries in terms of Mean Reciprocal Rank (MRR) and Mean Position (MEANPOS) of the first relevant URL output by the search engine. Table 4.3 shows the MRR scores for each algorithm with popular bookmark queries. The best result was achieved when combining all the spam detection methods proposed, with an improvement of 26.98% in MRR when compared to PageRank. The last column shows the T-test results, which indicate the statistical significance of the difference in results[19] for

[19]Recall that statistical significance is not computed on the average result itself, but on each evaluation evidence (i.e., it also considers the agreement between subjects when assessing the

Method	MRR	Gain [%]	Signific., p-value
ALL LINKS	0.3781	-	-
UMSR	0.3768	-0.53%	*No*, 0.34
BMSR	0.4139	9.48%	*Highly*, 0.008
SLAbS	0.4141	9.5%	*Yes*, 0.04
SLLA	0.4241	12.14%	*Yes*, 0.03
BMSR+SLAbS	0.4213	11.40%	*Yes*, 0.02
SLLA+BMSR	0.4394	16.20%	*Highly*, 0.01
SLLA+SLAbS	0.4544	20.17%	*Highly*, 0.003
SLLA+BMSR+SLAbS	**0.4802**	**26.98%**	***Highly*, 0.001**

Table 4.3: Mean Reciprocal Rank (higher is better) for popular bookmark queries.

Method	MEANPOS	Gain [%]	Significance
ALL LINKS	6.35	-	-
UMSR	6.25	1.57%	*No*, 0.34
BMSR	5.37	18.25%	*Yes*, 0.04
SLAbS	5.84	8.72%	*No*, 0.26
SLLA	5	27.06%	*Highly*, 0.003
BMSR+SLAbS	5.63	12.89%	*Minimal*, 0.12
SLLA+BMSR	4.84	31.17%	*Highly*, 0.01
SLLA+SLAbS	4.68	35.86%	*Highly*, 0.002
SLLA+BMSR+SLAbS	**4.62**	**37.29%**	***Yes*, 0.04**

Table 4.4: Mean position of the first relevant result obtained for popular bookmark queries.

each database when compared to the ALL LINKS version (i.e., PageRank on the original link database). The only method that had a negative impact on MRR was the UMSR, which indicates that many unidirectional relations between sites are rather useful for the ranking (i.e., not artificial). This was also the only algorithm for which the T-test indicated a non-significant difference in the results (p-values lower than 0.25 are taken as marginally significant, lower than 0.05 are taken as significant, and lower than 0.01 as highly significant).

Table 4.4 presents the Mean Position of the first relevant result (MEANPOS) achieved for popular bookmark queries under each of the algorithms we proposed. The best combination remains *SLLA+BMSR+SLAbS*, with a gain of 37.00%. Thus, we conclude that for popular bookmark queries the combination of all

results). Thus, smaller average differences could result in a very significant result, if the difference between the two algorithms remains relatively constant for each subject.

Method	**MRR**	**Gain [%]**	**Signific., p-value**
ALL LINKS	0.3200	-	-
UMSR	0.3018	-5.68%	*Highly*, 0.01
BMSR	0.3195	-0.17%	*No*, 0.45
SLAbS	0.3288	2.73%	*No*, 0.31
SLLA	0.3610	12.81%	*Yes*, 0.04
BMSR+SLAbS	0.3263	-2.19%	*No*, 0.36
SLLA+BMSR	0.3632	13.47%	*Yes*, 0.03
SLLA+SLAbS	0.3865	20.78%	*Yes*, 0.017
SLLA+BMSR+SLAbS	**0.3870**	**20.92%**	***Yes*, 0.016**

Table 4.5: Mean Reciprocal Rank (higher is better) for randomly selected bookmark queries.

methods is the best spam detection solution. Also, individually, Site Level Link Alliance (SLLA) produced the highest increase in PageRank quality.

After having evaluated our techniques on popular bookmark queries, we tested their performance over the randomly selected ones. The MRR results for this scenario are displayed in Table 4.5. Again, the best outcome was achieved when combining all the spam detection methods proposed, with an improvement of 20.92% in MRR when compared to PageRank. Note that an improvement is harder to achieve under this setting, since the Web pages searched in these queries are not necessarily popular, and thus many of them may have just a few in-going links and consequently a low PageRank score. Therefore, as removing links at the site level might also have the side effect of a further decrease of their PageRank score, they could become even more difficult to find. This is why both site level mutual reinforcement algorithms (BMSR and UMSR) resulted in a negative impact in the results, indicating that *some* site level mutual reinforcement might not necessarily be a result of spam (at least the uni-directional type of reinforcement). Similar results have been observed when computing the Mean Position of the first relevant result, instead of the MRR (see Table 4.6). Individually, SLLA is still the best algorithm, whereas the best technique overall is again the combined SLLA+BMSR+SLAbS.

Topic Queries. As mentioned earlier in this section, we evaluated the topic queries using precision at the top 5 results (P@5) and at the top 10 results (P@10), as well as the mean average precision (MAP). We first turn our attention to the experiment in which the output URLs assessed both as relevant and highly relevant are considered as good results. Table 4.7 presents the evaluation for the most popular 30 topic queries under this scenario. All results were tested

Method	MEANPOS	Gain [%]	Significance
ALL LINKS	8.38	-	-
UMSR	8.61	-2.71%	*Highly*, 0.01
BMSR	8.28	1.28%	*No*, 0.27
SLAbS	8.23	1.80%	*Minimal*, 0.24
SLLA	7.42	12.89%	*Minimal*, 0.11
BMSR+SLAbS	8.02	4.09%	*No*, 0.36
SLLA+BMSR	7.27	15.21%	*Minimal*, 0.07
SLLA+SLAbS	7.12	17.61%	*Highly*, 0.01
SLLA+BMSR+SLAbS	**7**	**19.76%**	***Highly*, 0.005**

Table 4.6: Average mean position of the first relevant result for randomly selected bookmark queries.

for significance, and in both P@5 and P@10 no method manifested a significant gain or loss. Even so, in both P@5 and P@10 we see that BMSR has a slight gain over UMSR. SLLA exhibited the greatest gain in P@5, but the results were relatively similar for all algorithms in P@10. As for MAP, most of the results (except for SLAbS, BMSR, and their combination) had significant gain on MAP, when compared with the original link database. Finally, SLAbS performance was rather poor. However, this behavior of SLAbS was recorded only with this kind of queries, where it is also explainable: Some very popular sites might indeed get an abnormal support from several of their fans; some would consider this as spam, but our testers apparently preferred to have the ranks of these sites boosted towards the top. The best individual method was SLLA and the best combination was SLLA with BMSR, which was better than the combination of all three methods due to the negative influence of SLAbS.

The same experiment was then performed for the 30 randomly selected topic queries. Its results are depicted in Table 4.8. Here, SLLA remains a very effective individual algorithm, but SLAbS shows even better results. This indicates that an abnormal support for less popular sites usually appears as a result of spam. More, due to this special behavior of our algorithms, under this setting the main contributor to the combined measures was SLAbS, thus yielding the best MAP score for BMSR+SLAbS.

Before concluding this analysis, we also measured the quality of our methods under the same setting, but considering only the highly relevant output URLs as good results (recall that our subjects evaluated each URL as irrelevant, relevant and highly relevant for each query). For the popular topic queries (Table 4.9), the performance of the individual methods was similar to the scenario that considered

Method	P@5	P@10	MAP	Signif. for MAP
ALL LINKS	0.255	0.270	0.198	-
UMSR	0.255	0.282	0.207	*Highly*, 0.0031
BMSR	0.260	0.285	0.198	*No*, 0.3258
SLAbS	0.226	0.262	0.185	*Minimal*, 0.0926
SLLA	0.275	0.270	0.227	*Highly*, 0.0030
BMSR+SLAbS	0.226	0.276	0.200	*No*, 0.3556
SLLA+SLAbS	0.245	0.255	0.216	*Yes*, 0.0429
SLLA+BMSR	**0.270**	**0.273**	**0.231**	***Highly*, 0.0031**
SLLA+BMSR+SLAbS	0.245	0.259	0.223	*Yes*, 0.0129

Table 4.7: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *all* the relevance judgments for popular topic queries.

Method	P@5	P@10	MAP	Signif. for MAP
ALL LINKS	0.412	0.433	0.311	-
UMSR	0.442	0.442	0.333	*Highly*, 0.0030
BMSR	0.400	0.445	0.314	*No*, 0.3357
SLAbS	0.436	0.458	0.340	*Yes*, 0.0112
SLLA	0.461	0.455	0.327	*Yes*, 0.0125
BMSR+SLAbS	0.448	**0.470**	**0.358**	*Highly*, 0.0012
SLLA+BMSR	**0.485**	0.448	0.326	*Highly*, 0.0006
SLLA+SLAbS	0.461	0.461	0.354	*Minimal*, 0.0618
SLLA+BMSR+SLAbS	0.461	0.467	0.346	*Highly*, 0.0002

Table 4.8: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *all* the relevance judgments for random topic queries.

both relevant and highly relevant results, with the main difference being that here SLAbS gains about 12% over the original database, instead of losing. This is because the sites previously discovered due to spam or noisy links (i.e., those being very popular, but also abnormally supported by some fans) were considered only relevant by our testers, and thus not included in this more strict experiment. Finally, for the randomly selected queries (Table 4.10), SLAbS again showed the best individual performance (just as in the sibling experiment considering both kinds of relevance judgments), with the overall top scores being achieved for SLLA+BMSR+SLAbS and BMSR+SLAbS.

Conclusion. In order to make our results more clear, we also plotted their relative gain over regular PageRank (i.e., without spam detection). Figure 4.10 depicts this gain in percentage for bookmark queries and Figure 4.11 depicts it

Method	P@5	P@10	MAP	Signif. for MAP
ALL LINKS	0.152	0.141	0.112	-
UMSR	0.152	0.147	0.131	*Highly*, 0.0002
BMSR	0.152	0.150	0.127	*Highly*, 0.0022
SLAbS	0.152	0.147	0.126	*Yes*, 0.0172
SLLA	**0.162**	0.153	0.163	*Highly*, 0.00003
BMSR+SLAbS	0.152	**0.156**	0.128	*Highly*, 0.0016
SLLA+SLAbS	0.157	0.147	0.175	*Highly*, 0.00002
SLLA+BMSR	0.157	0.153	0.168	*Highly*, 0.00005
SLLA+BMSR+SLAbS	0.157	0.150	**0.179**	*Highly*, 0.00001

Table 4.9: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *only the highly relevant* results selected by our subjects for popular topic queries.

for topic queries. We first note that UMSR yielded negative results in three of the four experiments with bookmark queries, which makes it less preferable to its sibling BMSR, even though it performed better than the latter one with topical queries. Also, we observe that SLAbS performed quite well under both broad experimental settings, but SLLA is clearly the best single approach for bookmark queries. Finally, all combined measures performed very well, with SLLA+BMSR+SLAbS being the best one.

Practical Issues

Amount of removed links. Even though the amount of removed links does not necessarily represent the performance increase of each algorithm, it is still interesting to see how much did they trim the original link structure. We thus present these values in Table 4.11 (recall that SLLA does not remove any links, but only downgrades them). We observe that BMSR has removed a relatively low amount of links (at least when compared to the other methods), which indicates that SLLA+SLAbS could be preferred in practical implementations when faster computations of the algorithm are desired, at the cost of a minimally lower output quality.

Scalability. Algorithms dealing with large datasets as the Web need to have a very low complexity in order to be applied in a real environment. We argue that all the algorithms we proposed in this section have a computational cost growth linear in the number of pages.

Both Mutual Reinforcement detection algorithms behave in a similar way,

Method	P@5	P@10	MAP	Signif. for MAP
ALL LINKS	0.170	0.179	0.187	-
UMSR	0.176	0.191	0.196	*Yes*, 0.0457
BMSR	0.170	0.185	0.195	*Minimal*, 0.0520
SLAbS	0.182	0.191	0.201	*Yes*, 0.0200
SLLA	0.164	0.185	0.194	*No*, 0.2581
BMSR+SLAbS	0.188	0.197	0.207	*Highly*, 0.0068
SLLA+BMSR	0.182	0.194	0.205	*Highly*, 0.0090
SLLA+SLAbS	0.182	0.206	0.203	*Yes*, 0.0180
SLLA+BMSR+SLAbS	**0.200**	**0.212**	**0.208**	*Highly*, 0.0012

Table 4.10: Precision at the first 5 results, at the first 10 results, and Mean Average Precision considering *only the highly relevant* results selected by our subjects for random topic queries.

Method	Links Detected	% of Total Links
UMSR	9371422	7.16%
BMSR	1262707	0.96%
SLAbS	21205419	16.20%
UMSR+BMSR	9507985	7.26%
BMSR+SLAbS	21802313	16.66%

Table 4.11: Amount of links removed by each of our algorithms.

with UMSR being slightly less expensive than BMSR. The former one needs a simple pass over all links and thus has the complexity $O(|E|)$. If $M = Average_{p \in V}(Out(p))$, and if the in-links information is present in the search engine database, but with the in-links in a random order, then the complexity of BMSR is $O(|V| \cdot M^2)$, with M^2 being the cost of sequential searching. Furthermore, if the in-links are sorted, then the complexity falls to $O(|V| \cdot M \cdot \log M)$.

SLAbS is very similar to UMSR. For each page p we update the statistics about its in-going links. Thus, if $P = Average_{p \in V}(In(p))$, then the computational complexity of SLAbS is $O(|V| \cdot P)$.

SLLA is based on the in-linkers of a page p that are not from the same site as p. Thus, the algorithm needs to calculate the amount of links from pages from $In(p)$ that point to other pages within $In(p)$. If the out-links or the in-links are already sorted, the complexity of this approach is $O(|V| \cdot M^2 \cdot \log M)$. Otherwise, the complexity is $O(|V| \cdot M^3)$, since a sequential search is needed.

Finally, we note that all algorithms we proposed in this section do a page-by-page processing, thus being trivially parallelizable.

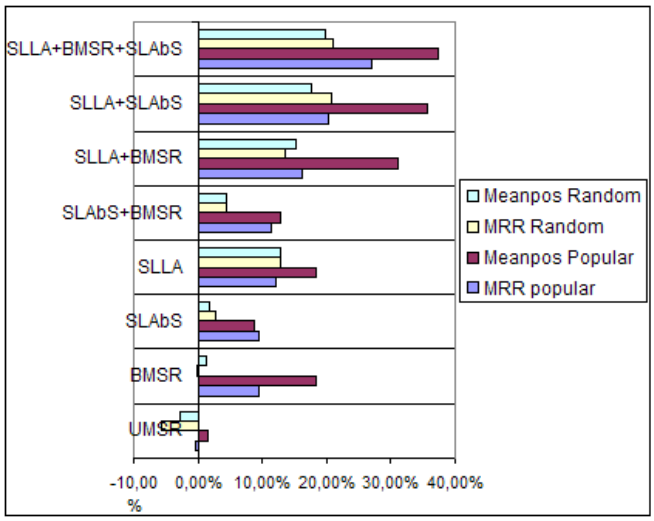

Figure 4.10: Relative gain (in %) of each algorithm in MRR and Mean Position for bookmark queries.

4.4.3 Discussion

There is no doubt that Search Engine Optimization is a really popular job right now. Just to give an example, as of writing this thesis, only in New York city there were several thousands SEO job openings[20]. This effect occurred because of two reasons: First, an efficient manipulation of the search engine rankings can bring millions of dollars into the target company; Second, even though such incorrect techniques are visibly decreasing the quality of our everyday search experience, they have not been explicitly moved outside the law. Thus, everybody can do it, as long as he is not detected / rank penalized by the search engine itself.

In this section we made another step in this continuous battle of keeping seach quality at very high standards. We introduced a novel approach to remove artificial linkage patterns from search engine databases. More specifically, we proposed to utilize site level link analysis to detect such malicious constructs. We designed and evaluated algorithms tackling three types of inappropriate site level relationships: (1) mutual reinforcement, (2) abnormal support and (3) link alliances. Our experiments have showed a quality improvement of 26.98% in Mean Reciprocal Rank for popular bookmark queries, 20.92% for randomly selected bookmark queries, and up to 59.16% in Mean Average Precision for topic queries.

Another important contribution we brought relates to the generality of our methods: Even though our main interest was "spam" detection, the algorithms we

[20]Information aggregated from several job sites.

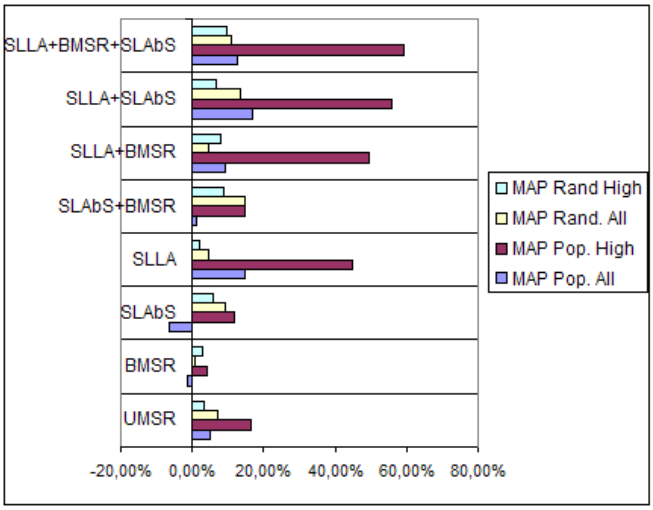

Figure 4.11: Relative gain (in %) in MAP for all algorithms for topic queries, considering only highly relevant results as relevant (High), and considering both relevant and highly relevant answers as relevant (All).

presented in this section can in fact detect much more types of artificial links. Thus, they are able to identify also intensively interlinked company branches, or site replicas, etc., all of which have not necessarily been intentionally deployed with the intent of manipulating search engine rankings. All in all, about 16.7% of the links from our collection were marked as noisy (i.e., non-votes), and quite a few of them could not necessarily be considered as nepotistic.

While most of the Web spam detection algorithms we presented in this section remove the identified malicious links completely, in the future one could investigate using different weights for various types of links, according to the relation they represent (i.e., inter-site or intra-site relation), as well as to their probability of representing a vote of importance. Finally, it would also be beneficial to design more complex (eventually automatic) approaches to tune up the parameter thresholds.

Chapter 5

Ranking for Web Search Personalization

5.1 Introduction

We have seen in the previous two chapters that link analysis ranking can do a good work in many application areas for which it was previously either completely unexplored, such as Desktop search and Email spam detection, or partially explored, such as Web spam detection. This chapter comes to propose link analysis solutions for a more intensively investigated area, Personalized Web Search.

The booming popularity of Web search engines has determined simple keyword search to become the only widely accepted user interface for seeking information over the Web. Yet keyword queries are inherently ambiguous. The query "canon book" for example covers several different areas of interest: religion, digital photography, literature, and music[1]. Interestingly, this is one of the examples in which the first ten Google results do not contain *any* URL on the last topic. Clearly, search engine output should be filtered to better align the results with the user's interests. Personalization algorithms accomplish this goal by (1) learning / defining user's interests and (2) delivering search results customized accordingly: pages about digital cameras for the photographer, religious books for the clergyman, and documents on music theory for the performing artist. A recent study

[1]In music, a *canon* is a composition in which two (or more) voices sing the same melodic line but start at different moments in time.

presented by SearchEngineWatch [203] indicated that more than 80% of the users would prefer to receive such personalized search results.

In this chapter we propose to exploit manually created large scale information repositories to personalize Web search, i.e., to return search results which are relevant to the user profile and are of good quality. There are two different types of such repositories: (1) Globally edited ones and (2) individually edited ones. For the former category we build on top of the metadata accumulated within public taxonomies such as the Open Directory. We thus project link analysis ranking into a taxonomical space and define appropriate similarity measures to order the Web search output in accordance to each user's preferences. Furthermore, we also describe a new algorithm that learns user profiles based on projecting search statistics on top of the same taxonomical space, thus being either faster, or less obtrusive than its predecessor approaches.

For the category of individually created repositories we propose to expand user's Web queries by utilizing both text and link analysis on top of Personal Information Repositories[2]. Query expansion assists the user in formulating a better query by appending additional keywords to the initial search request in order to encapsulate her interests therein, as well as to focus the search output accordingly. The technique has already been shown to perform very well when used over large data sets and especially with short input queries [147, 40]), i.e., exactly under the characteristics of the Web search scenario. Therefore, we automatically extract additional keywords related both to the user's search request and to her interests as captured by her PIR, thus implicitly personalizing the search output.

The chapter is organized as follows: We proceed with a discussion of the background literature specific for this area. Further, we discuss how to exploit global repositories in order to personalize search in Section 5.3, as well as to automatically infer user profiles in Section 5.4. The use of individually created collections is then presented in Section 5.5. The chapter ends with a flexibility analysis relevant for our both personalization approaches, followed by some conclusions and an outline of possible future directions.

5.2 Specific Background

Personalization can bring extensive improvements over regular search quality. And yet, even though many authors attempted to design a widely accepted personalized search algorithm, no one has succeeded so far. This section reviews most

[2]Recall that the PIR is defined as the personal collection of textual documents, emails, cached Web pages, etc. stored on a local machine.

techniques that have been proposed along the time and compares them to the two approaches we introduce in this chapter. Moreover, since our second algorithm builds on top of another existing domain, namely automatic query expansion, we also survey this research area in more detail within the second half of the section.

5.2.1 Personalized Search

Introduction. Search personalization consists of two highly interacting components: (1) user profiling, and (2) output ranking. Generally, the latter one depends on the characteristics of the former: A good profile definition method is needed before designing the ranking algorithm. Thus, user profiles represent a central point of all approaches to personalize search. In one way or another, one must know the profile of the person searching the Web in order to deliver the best results. We distinguish two possibilities for classifying the algorithms for personalized search, both centered around the user profile: The first relates to the way it is *constructed*, whereas the second relates to the way it is *used*. Most works available so far are focused on the former approach: Chan [48] for example investigated the types of information available to pursue it, such as the time spent visiting a Web page, or the frequency of visits, whether it was bookmarked, etc. Since many authors attempted to exploit this kind of information over the past few years, Kelly and Teevan [138] have recently put together a comprehensive bibliography capturing and categorizing these techniques. In this section we will present a different literature survey, focused on the latter perspective, i.e., the way profiles are used to achieve personalization, rather than on the way they are built. We distinguish three broad approaches: (1) integrating the personalization aspect directly into PageRank, (2) filtering each query output to contain only URLs relevant to the user profile, and (3) using the personalized search algorithm as an additional search engine measure of importance (together with PageRank, TFxIDF, etc.). Let us now inspect each of them in detail.

PageRank-based Methods. The most efficient personalized search algorithm will probably be the one which has the personalization aspect already included in the initial rankings. Unfortunately, this seems very difficult to accomplish. Initial steps in this direction have been already described by Page et al. [172], who proposed a slight modification of the PageRank algorithm to redirect the random surfer towards some preferred pages. However, it is clearly impossible to compute one PageRank vector for each user profile, i.e., for each set of pages "privileged" by the random surfer. The same is valid when the random surfer is uniformly choosing pages to jump to and some preferred domains (or links) get a higher weight during the computation [5, 72].

Qiu and Cho [176] used Machine Learning to classify user's preferences into one or more top level ODP topics, and then applied Topic-Sensitive PageRank [118] (see also Section 2.1.2 for its complete description) weighted according to the inferred user interests.

Jeh and Widom [131] proposed an algorithm that avoids the huge resources needed for storing one Personalized PageRank Vector (PPV) per user. They started from a *set of hubs* (H)[3], each user having to select her *preferred pages* from it. PPVs can then be expressed as a linear combination of PPVs for preference vectors with a single non-zero entry corresponding to each of the pages from the preference set (called basis vectors). Furthermore, basis vectors are decomposed into partial vectors (encoding the part unique to each page, computed at run-time) and the hubs skeleton (capturing the interrelationships among basis vectors, stored off-line). The advantage of this approach is that for a hub set of N pages, one can compute 2^N Personalized PageRank vectors without running the algorithm again. These rankings are generated off-line, independently of the user query, which does not impose any additional response time on the search engine. The disadvantages are the necessity for the users to select their preference set only from within a given group of pages[4] (common to all users), as well as the relatively high computation time for large scale graphs. The latter problem has been solved through several subsequent studies [101, 190], their most recent solution using rounding and count-min sketching in order to fastly obtain accurate enough approximations of the Personalized PageRank scores.

Output Filtering Methods. As partially or completely query independent techniques still exhibit a number of limitations, query oriented approaches have been investigated as an alternative. One of them is to sort out the irrelevant or likely irrelevant results, usually in a process separated from the actual ranking mechanism. Liu et al. [163] restrict searches to a set of categories defined in the ODP (via Google Directory). Their main contribution consists of investigating various techniques to exploit users' browsing behavior for learning profiles as bags of words associated to each topical category. In comparison to our approaches, they use relatively time consuming algorithms (e.g., Linear Least Squares Fit) and obtain a slightly worse precision. A different scheme is presented by Pahlevi and Kitagawa [173], where for each query the user first selects her topics of interest, and then a classifier is used to either mark the results as non-relevant, or associate them to one of the specified categories. Similarly, in [140] users start

[3]Recall that hubs were defined here as pages with high PageRank, differently from the more popular definition of Kleinberg [143].

[4]We have done some work in the direction of improving the quality of this set of pages (see Chirita et al. [73]), but users are still restricted to select their preferred pages from a subset of H (if $H = \{CNN, FOXNews\}$ we cannot bias on MSNBC for example).

by building a concept tree and then select one of these concepts to search for. The output is constructed by generating a set of queries describing user's selected concept(s) and combining their results. Both these latter approaches are very difficult to accomplish on a Web scale, either because classification delays search engine response time too much, or because users are not willing to define concept hierarchies every time they search. Finally, when utilized in conjunction with specific user profiles, automatic query expansion represents a different kind of output filtering, its main underlying idea being to focus the search output onto the additional keywords appended to the query. We refer the reader to the next subsection for a detailed discussion on the currently existing query expansion techniques.

Re-ranking Methods. A third approach to achieve personalization relies on building an independent simpler personalized ranking algorithm, whose output is then combined with that of PageRank. Sugiyama et al. [202] analyze user's surfing behavior and generate user profiles as features (terms) of the pages they visited. Then, upon issuing a new query, the results are ranked based on the similarity between each URL and the user profile. Similarly, Gauch et al. [107] build profiles by exploiting the same surfing information (i.e., page content and length, time spent on each URL, etc.), as well as by spidering the URLs saved in the personal Web cache and classifying them into topics of interest (in a more recent work [199], they log this latter information using a Google wrapper instead). Both approaches are similar to our taxonomy based personalized search algorithm, but we construct user profiles only by analyzing the user queries submitted to the search engine, rather than the entire browsing behavior, thus being less intrusive and allowing these data to be collected directly on the server side – e.g., via search accounts such as Yahoo! My Web, while having a faster ranking scheme.

Besides the algorithm depicted in Section 5.5, there exists only one other approach attempted to enhance Web search using Desktop data. Teevan et al. [204] modified the query term weights from the BM25 weighting scheme [132] to incorporate user interests as captured by their Desktop indexes. The method is orthogonal to our work, since we apply query expansion, a personalization tool much more powerful than term weighting.

Commercial Approaches. On the one hand, industry has long claimed that personalization distinguishes itself as one of the future technologies for Web search. On the other hand, none of the techniques initiated by the search engines in this direction has managed to succeed in being widely accepted by the public. This indicates that more work needs to be performed before identifying the best personalization approach.

Most major search engines offer personalization services as beta services. Google

used to ask users about Open Directory topics of interest in order to achieve personalization[5], possibly by implementing an extension of Haveliwala's Topic-Sensitive PageRank [118]. This prototype is currently no longer available. Its replacement is Google Search History[6], which has the additional advantage of learning the user profile based on her previous queries and clicks. Yahoo! and Ask offer quite similar services respectively via their MyWeb 2[7] and MyJeeves[8] applications.

A different approach is taken by Eurekster[9], which offers personalization based on user communities, i.e., by exploiting the interests of users with interests closely related to those of the target user. Other major search engines might apply this kind of community based query log mining as well, including for example Windows Live Search[10]. Some of the meta-search engines also claim to have access to personalization data (see for example IBoogie[11]) when aggregating the results over different search providers.

JetEye[12] was one of the first search engines to provide user specific customization of the core search engine, by deciding what should be tapped for the future, what should be excluded, etc. Other implementations of this approach exist, for example within Google Coop[13], Rollyo[14], or LookSmart's Furl[15], etc. The new aspect therein is that personalization is also included in the index collection by adding specific Websites to it, comments, keywords, or quality ratings.

A lot of other personalization search engines exist, such as Memoory[16], or A9[17]. Also, other companies attempting Web search personalization include Kaltix or Outride, both bought by Google in 2003 and 2001 respectively.

SnakeT[18] [97] includes some naïve form of personalization, according to which users are able to select some clusters of interest, once the output of their queries has been grouped into categories.

[5]http://labs.google.com/personalized
[6]http://www.google.com/psearch?hl=en
[7]http://myWeb2.search.yahoo.com
[8]http://myjeeves.ask.com/
[9]http://search.eurekster.com
[10]http://search.live.com
[11]http://www.iboogie.com
[12]http://www.jeteye.com/
[13]http://www.google.com/coop
[14]http://www.rollyo.com/
[15]http://www.furl.net/
[16]http://www.searchenginejournal.com/?p=1179
[17]http://www.a9.com/
[18]http://www.snaket.com/

5.2.2 Automatic Query Expansion

Automatic query expansion aims at deriving a better formulation of the user query in order to enhance retrieval. It is based on exploiting various social or collection specific characteristics in order to generate additional terms, which are appended to the original input keywords before identifying the matching documents returned as output. In this section we survey some of the representative query expansion works grouped according to the source employed to generate additional terms: (1) Relevance feedback, (2) Collection based co-occurrence statistics, and (3) Thesaurus information. Some other approaches are also addressed in the end of the section.

Relevance Feedback Techniques. The main underlying idea of Relevance Feedback (RF) is that useful information can be extracted from the relevant documents returned for the initial query. First approaches were manual (and therefore personalized) [185] in the sense that the user was the one choosing the relevant results, and then various methods were applied to extract new terms, related to the query and the selected documents. Efthimiadis [90] presented a comprehensive literature review and proposed several simple methods to extract such new keywords based on term frequency, document frequency, etc. We used some of these as inspiration for our Desktop specific expansion techniques. Chang and Hsu [49] asked users to choose relevant clusters, instead of documents, thus reducing the amount of user interaction necessary. Yet RF has been shown to be effectively automatized by simply considering the top ranked documents as relevant [219] (this technique is known as Pseudo RF). Lam and Jones [152] used summarization to extract informative sentences from the top-ranked documents, and appended them to the user query. We have adapted this approach for our Desktop scenario. Also, Carpineto et al. [42] maximized the divergence between the language model defined by the top retrieved documents and that defined by the entire collection. Finally, Yu et al. [221] selected the expansion terms only from vision-based segments of Web pages in order to cope with the multiple topics residing therein.

Co-occurrence Based Techniques. Another source of additional query terms is the searched collection itself. Terms highly co-occurring with the originally issued keywords have been shown to increase precision when appended to the query [141]. Many statistical measures have been developed to best assess "term relationship" levels, either based on analyzing the entire documents [177], lexical affinity relationships [40] (i.e., pairs of closely related words which contain exactly one of the initial query terms), etc. We have investigated three such approaches in order to identify query relevant keywords from the rich, yet rather complex Personal Information Repository. In a more recent investigation, Wang and Tanaka

[212] first employed a topical based clustering of all terms, and then selected the candidate expansion words using conditional entropy.

Thesaurus Based Techniques. A broadly explored technique is to expand the user query with new terms, whose meaning is closely related to the original input keywords. Such relationships are usually extracted from large scale thesauri, as WordNet [167], in which various sets of synonyms, hypernyms, hyponyms, etc. are predefined. Just as for the term co-occurrence methods, initial experiments with this approach were controversial, either reporting improvements, or even reductions in the output quality (see for example the work of Voorhees [209] and the references therein). Recently, as the experimental collections grew larger, and as the employed algorithms became more complex, better results have been obtained. Shah and Croft [194], as well as Kim et al. [142], applied various filtering techniques over the proposed additional keywords, either by estimating query clarity [77], or by using a root sense tagging approach. We also use WordNet based expansion terms. However, we extend this process with an analysis of the Desktop level relationship between the original query and the proposed additional keywords.

Other Techniques. There exist several other attempts to extract better terms for query expansion, two of them being specifically tailored for the World Wide Web environment: Cui et al. [78] generated word correlations utilizing a new probability for query terms to appear in each document, computed over the search engine logs. Kraft and Zien [147] pointed out that anchor text is very similar to the user queries, and thus exploited it to acquire additional keywords. Both approaches are orthogonal to our Desktop focused work, as we use a different and richer source of expansion terms.

5.3 Taxonomy Based Personalized Web Search

We presented in Section 5.2.1 the most popular approaches to personalizing Web search. Even though they are the best so far, they all have some important drawbacks. PageRank based methods either need too many resources to compute and store all Personalized PageRank Vectors, or are limited to a very restricted set of pages to personalize on. Existing re-ranking methods usually employ text classifiers both for learning user profiles and for evaluating query results (i.e., URLs), thus being inherently slow. Finally, output filtering schemes must perform an additional results trimming step, which is also delaying response time to some extent. It is therefore still worth searching for a simpler and faster algorithm with at least similar personalization granularity as the current ones.

In this section we propose to use community wide manually entered catalogue metadata, such as the ones collected within the Open Directory, which express topical categorizations of Web pages. This kind of metadata was one of the first available on the Web in significant quantities (for example within Yahoo! Directory), providing hierarchically structured access to high-quality content on the Web. We thus build upon the categorization done in the context of the ODP, as it is one of the largest efforts to manually annotate Web pages. Over 74,000 editors are busy keeping the ODP directory reasonably up-to-date, delivering access to over 5 million Web pages in its catalogue. However, its advanced search offers a rudimentary "personalization" feature by restricting search to the entries of just one of the 16 main categories. Google Directory (which is also built on top of ODP) provides a related feature, by offering to restrict search to a specific category or subcategory. Clearly these services yield worse results than searching Google itself [67]. Can we improve them, taking user profiles into account in a more sophisticated way, and how will these enhanced personalized results compare to the ordinary Google results? To what extent will they be helpful? More specifically, will they improve Google for all kinds of queries (including very exact queries having, e.g., five words or more), only for some queries, or not at all? In the following section, we analyze and propose answers to these questions, and then we evaluate them experimentally in Section 5.3.3.

5.3.1 Algorithm

Our search personalization algorithm exploits the annotations accumulated in generic large-scale taxonomies such as the Open Directory. Even though we concentrate our forthcoming discussion on ODP, *any* similar taxonomy can be used. We define *user profiles* taking a simple approach: each user has to select several topics from ODP, which best fit her interests. For example, a user profile could look like this:

/Arts/Architecture/Experimental
/Arts/Architecture/Famous_Names
/Arts/Photography/Techniques_and_Styles

At run-time, the output given by a search service (from MSN, Yahoo!, Google, etc.) is re-sorted using a calculated (link) *distance* from the user profile to each output URL. This translates into a minimal additional overhead to the search engine, bounded by the time needed to include the above mentioned distances into the overall ranking scheme. The execution is depicted in Algorithm 5.3.1.1.

Algorithm 5.3.1.1. Personalized Search.

Input: $Prof_u$: Profile for user u, given as a vector of topics
Q : Query to be answered by the algorithm.
Output: Res_u: Vector of URLs, sorted after user u's preferences

1: Send Q to a search engine S (e.g., Google)
2: Res_u = Vector of URLs, as returned by S
3: **For** $i = 1$ **to Size**(Res_u)
$Dist[i] = Distance(Res_u[i], Prof_u)$
4: **Sort** Res_u using $Dist$ as comparator

We additionally need a function to estimate the distance between a URL and a user profile. Let us inspect this issue in the following discussion.

5.3.2 Estimating Topic Similarity

When performing search on Open Directory, each URL comes with an associated ODP topic. Similarly, many of the URLs output by Google are connected to one or more topics within the Google Directory (almost 50% of Top-100, as we observed in our experiments described in Chirita et al. [67]). In both cases, for each output URL we are dealing with two sets of nodes from the topic tree: (1) Those representing the user profile (set A), and (2) those associated with the URL (set B). The (link) distance between these sets can then be defined as the minimum (link) distance between all pairs of nodes given by the Cartesian product $A \times B$. There are quite a few possibilities to define the distance between two nodes, depending on the perspective we take on ODP: as a tree, as an ontology, as a graph, etc. In the following, we will present the most representative metrics we found suitable for our algorithm, following an increasing level of complexity.

Naïve Distance. The simplest solution is minimum tree-distance, which, given two nodes a and b, returns the sum of the minimum number of tree links between a and the subsumer (the deepest node common to both a and b) plus the minimum number of tree links between b and the subsumer (i.e., the shortest path between a and b). On the example from Figure 5.1, the distance between */Arts/Architecture* and */Arts/Design/Interior_Design/Events/Competitions* is 5, and the subsumer is */Arts*.

If we also consider the inter-topic links from the Open Directory, the simplest distance becomes the shortest path between a and b. For example, if there is a link

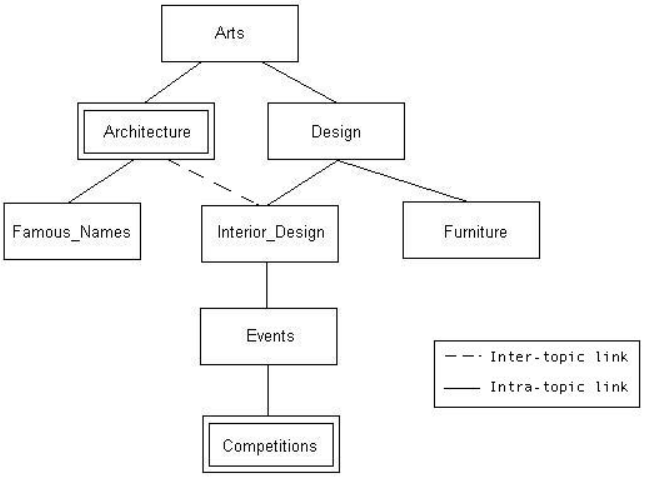

Figure 5.1: Example tree structure of topics from ODP

between *Interior_Design* and *Architecture* in Figure 5.1, then the distance between *Competitions* and *Architecture* is 3. This solution implies loading either the entire topic graph or all the inter-topic links into memory. Its utility is somewhat questionable: the existence of a link between *Architecture* and *Interior_Design* does not always imply that a famous architect (one level below in the tree) is very close to the area of interior design. We therefore chose to consider only the intra-topic links directly connected to a or b when computing the shortest path between them (similar to [164]).

Tree Similarity. The main drawback of the previous metric comes from the fact that it ignores the depth of the subsumer (lowest common ancestor). The bigger this depth is, the more related are the nodes (i.e., the concepts represented by them). This problem is solved using metrics from Information Theory [76], such as the semantic similarity between two topics t_1 and t_2 in a taxonomy [164]. This is defined as the ratio between their common meaning and their individual meanings, as follows:

$$\sigma_s^T(t_1, t_2) = \frac{2 \cdot \log Pr[t_0(t_1, t_2)]}{\log Pr[t_1] + \log Pr[t_2]} \tag{5.1}$$

where $t_0(t_1, t_2)$ is the subsumer of t_1 and t_2, and $Pr[t]$ represents the prior probability that any page is classified under topic t (computed for example as the ratio between the number of pages stored in the subtree rooted at node t and the overall amount of pages).

Concept Similarity. The ODP is manually annotated by human editors and it sometimes contains more URLs within a narrow, but popular concept, compared

to a wider, but less popular one. This can yield inconsistent results with our second metric, which motivated us to search for a more elaborate approach. Li et at. [162] investigated ten intuitive strategies for measuring semantic similarity between words using hierarchical semantic knowledge bases such as WordNet [167]. Each of them was evaluated empirically on a group of testers, the best one having a 0.9015 correlation between human judgment and the following formula:

$$S(a,b) = e^{-\alpha \cdot l} \cdot \frac{e^{\beta \cdot h} - e^{-\beta \cdot h}}{e^{\beta \cdot h} + e^{-\beta \cdot h}} \tag{5.2}$$

The parameters are as follows: α and β were defined as 0.2 and 0.6 respectively, h is the tree-depth of the subsumer, and l is the semantic path length between the two words. If we have several words attached to each concept and sub-concept, then l is 0 if the two words are in the same concept, 1 if they are in different concepts, but the two concepts have at least one common word, or the tree shortest path if the words are in different concepts which do not contain common words.

Although this measure is very good for words, it is still not perfect when we apply it to the Open Directory topical tree, because it does not make a difference between the distance from a (the profile node) to the subsumer, and the distance from b (the output URL) to the subsumer. Consider for example node a to be */Top/Games* and b to be */Top/Computers/Hardware/Components/Processors/x86*. A teenager interested in computer games (level 2 in the ODP tree) could be very satisfied receiving a page about new processors (level 6 in the tree), which might increase her gaming quality. On the other hand, the opposite scenario (profile on level 6 and output URL on level 2) does not hold any more, at least not to the same extent: a processor manufacturer will generally be less interested in the games existing on the market. This leads to our following extension of the above formula:

$$S'(a,b) = ((1-\gamma) \cdot e^{-\alpha \cdot l_1} + \gamma \cdot e^{-\alpha \cdot l_2}) \cdot \frac{e^{\beta \cdot h} - e^{-\beta \cdot h}}{e^{\beta \cdot h} + e^{-\beta \cdot h}} \tag{5.3}$$

with l_1 being the shortest path from the profile to the subsumer, l_2 the shortest path from the URL to the subsumer, and γ a parameter in $[0,1]$.

Graph Similarity. An even more complete similarity function can be found in [164]. It estimates the similarity between two topics based on several sources of information, using the following formula:

$$\sigma_s^G(t_1,t_2) = \max_k \frac{2 \cdot \min(W_{kt_1}, W_{kt_2}) \cdot \log Pr[t_k]}{\log\left(Pr[t_1|t_k] \cdot Pr[t_k]\right) + \log\left(Pr[t_2|t_k] \cdot Pr[t_k]\right)} \tag{5.4}$$

As in Information Theory, the probability $Pr[t_k]$ represents the prior probability that any document is classified under topic t_k, while $Pr[t_i|t_k]$ represents the posterior probability that any document will be classified under topic t_i, given that

it is classified under t_k. Finally, W_{ij} can be interpreted as a fuzzy membership[19] value of topic t_j in the subtree routed at t_i.

Even though this formula seems to promise good results, computing it is very resource demanding: The authors reported the use of over 5,000 CPU hours on a supercomputer facility to calculate the similarity values for a (large) subset of ODP. Since we do not have such a facility available, and since computing this measure on-line (i.e., at search time) is quite time consuming as well, we decided not to experiment with this measure.

Combining the Similarity Function with the Google Page Scores. If we use Google to do the search and then sort the URLs according to the Google Directory taxonomy, some high quality pages might be missed (i.e., those which are top ranked, but which are not in the directory). In order to integrate those, the above formulas can be combined with the existing Google score associated to each page. We propose the following approach:

$$S''(a,b) = \delta \cdot S'(a,b) + (1-\delta) \cdot GooglePageScore(b) \tag{5.5}$$

with δ being another parameter in $[0,1]$ allowing us to keep the final score $S''(a,b)$ inside $[0,1]$ (for normalized page rank scores). If a page is not in the directory, we take $S'(a,b)$ to be 0.

Observation. Human judgment is a non-linear process over information sources [162], and therefore it is very difficult (if not impossible) to propose a metric which is in perfect correlation to it. However, we think that the thorough experimental analysis presented in the next section will provide a good approximation of the utility of these metrics.

5.3.3 Experiments

Experimental Setup

To evaluate the benefits of our personalization algorithm, we interviewed 24 of our colleagues (researchers in different computer science areas, psychology, education and design), asking each of them to define a user profile according to the Open Directory topics (see Section 5.3.1 for an example profile), as well as to choose six queries of the following types:

- One single-word *specific* query, which they *thought to have one or maximum two meanings*[20] (for example "PageRank").

[19] It is fuzzy because it also incorporates the "related" and "symbolic" links from ODP.

[20] Of course, that did not necessarily mean that the query had no other meaning.

- One single-word *relatively ambiguous* query, which they thought to have two or three meanings (for example "latex").
- One single-word *ambiguous* query, which they thought to have at least three meanings, preferably more (for example "network").
- Three queries of the same types as above, but with *multiple keywords* (i.e., at least two, preferably more). Some examples could include "www Edinburgh 2006" for the ambiguous query, "pocket pc software" for the semi-ambiguous one, and "topcoder programming competitions" for the specific query. The average query length for these three multi-word query types turned out to be 2.34 in the case of the ambiguous queries, 2.50 for the semi-ambiguous ones, and 2.92 for the specific queries.

We compared the test results using the following nine approaches[21]:

1. **Google Search**, as returned by the Google API[22].
2. **Google with Naïve Distance**, using our algorithm from Section 5.3.1 to reorder the Top-100 URLs returned by the Google API, and having as input the Google Directory topics returned by the API for each resulting URL.
3. **Google with Tree Distance**, under the same setting as above, but using the tree distance instead.
4. - 6. **Google with Concept Similarity**, using three different values for the γ parameter: 0.6, 0.7, and 0.8[23].
7. - 9. **Combined PageRank with Concept Similarity**, using three values for the δ parameter: 0.3, 0.5, and 0.7[24].

[21]Note that we actually performed two experiments. The first one, presented in Chirita et al. [67], brought only the conclusion that Google with Naïve Distance is better than Google, which in turn is better than ODP search. The second one is much more comprehensive, and therefore described in this section in detail.

[22]http://api.google.com

[23]The parameters for both this algorithm and the subsequent one were first selected by asking 4 persons to perform Web searches utilizing all 9 parameter values from 0.1 to 0.9 at equal intervals. Their results were not analyzed in detail, but used only to choose suitable parameters for the actual evaluation process. This helped us to observe that one should weigh the distance between the profile and the subsumer more than the distance between each URL and the subsumer. We selected the best three values for γ, namely $\gamma \in \{0.6, 0.7, 0.8\}$, in order to find the exact ratio between these weights.

[24]Here we investigated whether PageRank is more important, or the concept similarity measure. As within the training phase the results were overall inconclusive, with some queries performing much better than others over various settings of the parameters, we chose to analyze $\delta \in \{0.3, 0.5, 0.7\}$, as a representative set of values for all possibilities (i.e., visibly more bias on the taxonomy, visibly more bias on the search engine, or equal). The only characteristic we observed from the initial tuning phase was that δ should be inversely proportional to the difficulty of the query, having values ranging from 0.1 to 0.9 (indeed, for some queries, the best

Query Cat.	$\gamma = 0.6$	$\gamma = 0.7$	$\gamma = 0.8$	**F-value [214]**
One-word	3.05	3.05	3.01	F(2,69,-) = 0.04
Multi-word	3.01	3.04	3.01	F(2,69,-) = 0.02

Table 5.1: Weights analysis for the concept similarity measure.

Query Cat.	$\delta = 0.3$	$\delta = 0.5$	$\delta = 0.7$	**F-value**
One-word	2.95	3.26	3.50	F(2,69,**99%**) = 5.44
Multi-word	3.53	3.23	3.07	F(2,69,**95%**) = 3.53

Table 5.2: Weights analysis for the combined ranking measure.

For each algorithm, each tester received the Top-5 URLs with respect to each type of query, 30 URLs in total. All test data was shuffled, such that testers were neither aware of the algorithm, nor of the ranking of each assessed URL. We then asked the subjects to rate each URL from 1 to 5, 1 defining a very poor result with respect to their profile and expectations (e.g., topic of the result, content, etc.) and 5 a very good one[25]. For each subset of 5 URLs we took the average grade as a measure of importance attributed to that $< algorithm, query\ type >$ pair.

Results

Having this dataset available, we performed an extensive analysis over it. First of all, we investigated the optimal values for the constants we defined. The results are summarized in Table 5.1 for the γ parameter and in Table 5.2 for δ. We decided to evaluate separately the single-word and the multi-word queries, since we expected them to perform differently. For γ, we found that all three values yielded similar results, the difference between their averages being far from statistically significant[26]. The outcome was much more interesting for δ: Not only were the results statistically significant, but they were also dependent on the query complexity. For simple queries it is better to use a large δ in order to give more weight to the Concept Similarity measure, whereas for more complex queries, PageRank should be made predominant through a small δ. The confidence level was 99% for the simple queries experiment and 95% for the one involving complex queries.

rating was obtained with $\delta = 0.1$ or 0.9 respectively).

[25]This is practically a weighted P@5, precision at the top 5 results.

[26]To evaluate the statistical significance of our experiments we used One-way and Two-way Analysis of Variance (ANOVA) [30].

Algorithm	Ambig. Q.	Semi-ambig. Q.	Specific Q.	Avg. / Algo.
Google	1.93	2.26	3.57	2.59
Naïve Distance	2.64, $p = 0.01$	2.75, $p = 0.04$	3.34	2.91, $p = 0.01$
Concept Sim.	2.67, $p < 0.01$	2.95, $p = 0.01$	3.52	3.05, $p < 0.01$
Combined	3.14, $p \ll 0.01$	3.27, $p \ll 0.01$	4.09, $p \ll 0.01$	3.50, $p \ll 0.01$
Avg. / Q. Type	2.60	2.80	3.63	

Table 5.3: Survey results for the one-word queries.

We then picked the best choices for the above mentioned parameters ($\gamma = 0.7$, which yielded the best results, though at a marginal difference; $\delta = 0.7$ for simple queries, which was significantly better than the other two investigated values with $p \ll 0.01$; and $\delta = 0.3$ for complex queries, also performing significantly best with $p \ll 0.01$) and measured the performance of each distance / similarity metric[27]. The overall results are depicted in Table 5.3 for single-word (simple) queries and in Table 5.5 for the multi-word (complex) ones, together with their p-values as compared to regular Google search.

In the simple queries scenario, all our proposed algorithms outperformed Google: The Naïve Distance received an average rating of 2.91 (out of 5, with $p = 0.02$), Concept Similarity received 3.05 ($p < 0.01$), and the Combined Measure reached 3.50 ($p \ll 0.01$), whereas Google averaged only 2.59. For the specific queries, Google managed to slightly surpass the ODP-based metrics (i.e, Naïve Distance and Concept Similarity), probably because for some of these queries ODP contains less than 5 URLs matching both the query and the topics expressed in the user profile. Even in this particular case, however, the Combined Metric yielded the best results (4.09 versus 3.57 for Google, a statistically significant difference with $p < 0.01$). As we expected, the more specific the query type was, the bigger its average rating (ranging from 2.60 for ambiguous queries up to 3.63 for the specific ones). The difference between ambiguous queries and semi-ambiguous ones was not statistically significant ($p = 0.13$), indicating that our subjects had difficulties separating these query types. However, there was a clear difference between the ratings for the semi-ambiguous queries and for the clear ones ($p < 0.01$). Thus, the distinction between ambiguity and clarity was easier to make. Overall, all these results were statistically significant at a 99% confidence level (details of the associated ANOVA analysis are depicted in Table 5.4). We noticed no interaction between the query type and the algorithm (last row of Table 5.4), which indicates that the average per algorithm is independent of the query type and vice-versa.

[27]Since the Tree Distance performed very close to the Naïve Distance, we decided not to include it in this analysis.

Src. of Var.	SS	Deg. of Free.	F-value
Algorithm	57.668	3	F(3,276,**99%**) = 9.893
Query Type	30.990	2	F(2,276,**99%**) = 27.614
Interaction	7.363	6	F(6,276,-) = 1.175

Table 5.4: Statistical significance analysis for the one-word queries experiments.

Algorithm	Ambig. Q.	Semi-ambig. Q.	Specific Q.	Avg. / Alg.
Google	2.32	3.19	4.15	3.22
Naïve Distance	2.17	2.95	3.60	2.90
Concept Sim.	2.40, $p = 0.31$	3.09	3.62	3.04
Combined	3.08, $p \ll 0.01$	3.75, $p < 0.01$	3.76	3.53, $p \ll 0.01$
Avg. / Q. Type	2.49	3.24	3.78	

Table 5.5: Survey results for the multi-word queries.

The results were tighter under the multi-word queries scenario (Table 5.5): Google average scores were somewhat above those of the Naïve Distance and Concept Similarity, the best metric being, as in the previous experiment, the Combined Measure (and that at a statistically significant difference with $p \ll 0.01$). For specific queries, Google performed best, but was strongly surpassed by the Combined Measure when using ambiguous and semi-ambiguous query keywords ($p < 0.01$ in both latter cases). The overall averages per algorithm were differentiated enough to achieve a 99% overall confidence level (see Table 5.6 for the ANOVA details), with a minimal interaction between the results associated to the query type and those associated to the algorithm type.

5.4 Taxonomy Based Automatic User Profiling

The algorithm we presented in the previous section builds on top of manually entered user profiles. This interaction step clearly adds some burden upon the users of our system and demands for automatized profiling techniques. In fact, this is a common problem for many on-line personalized applications, which is why researchers have attempted to automatically learn such user profiles usually by exploiting various aspects of browsing behavior (see for example [187]). In most cases this implied the analysis of either very personal or very many usage data. In this section we advance one step further: We learn user profiles only from past search engine queries, which is still a bit intrusive (about 15% of our prospective testers refused to participate in the experiments, arguing that their

Src. of Var.	SS	Deg. of Free.	F-value
Algorithm	16.011	3	F(3,276,**99%**) = 4.827
Query Type	80.547	2	F(2,276,**99%**) = 36.430
Interaction	9.450	6	F(6,276,**75%**) = 1.424

Table 5.6: Statistical significance analysis for the multi-word queries experiments.

queries are private), but clearly less intrusive than all previous approaches. We also need very little data to converge to a profile expressing user's interests well enough.

We start with the following observation: Since current personalized search algorithms either need extensive information to learn the user profile [202, 107], or simply have it entered manually, could we use ODP to provide a more accurate, less intrusive and less resource consuming method to learn user profiles? This section will provide an algorithm to answer this question positively.

5.4.1 Algorithm

Overview. Learning user profiles (also known as *Preference Elicitation* [50]) inherently needs some kind of personal user input data, such as (search engine) queries, bookmarks, time or frequency of visits to a set of Web pages, etc. We minimize the amount of these data by exploiting only queries sent to a search engine. For each query, we add to the user profile the weighted set of distinct ODP topics associated to its output URLs, as follows: Every time a topic appears in the result set, a weight of 1 is added; if a URL has no topic associated to it, we try to infer its topic by analyzing the topic(s) associated to its related pages (using the *Related Pages* feature of the Google API), to its home page, or to its parent directory. Since these latter topics have been automatically deducted, their weight is represented by a parameter smaller than 1. After a few days of surfing, the profile can be generated as the set of topics with the highest N weights[28].

Before testing the algorithm, we evaluated the feasibility of our solution to derive the topics associated to a URL missing from ODP. We selected 134 queries either used by previous researchers (e.g., [118]), or listed as most the frequent queries by Google[29] or Lycos[30]. Two experts rated the similarity between the topics

[28]Even the user profiles generated after one day of surfing do provide useful information about each user's interests. However, they also contain several false topical interest predictions, either because many of the queries issued that day cover several topics, or because the user issued some erroneous queries (e.g., with typos). Therefore, in general, several days of surfing are necessary.

[29]http://www.google.com/press/zeitgeist.html

[30]http://50.lycos.com/

associated to each URL and those associated to its related pages, to its home page, and to its parent directory. The grades were between 0 and 1, "zero" meaning a totally unrelated topic and "one" representing exactly the same topic. The average values we obtained were $\lambda = 0.64$ for the related URLs and $\mu = 0.46$ for the home pages (we dropped the analysis of the parent directory, because in almost all cases it did not output any topic). We then used these two values as a weight for the topics *heuristically* associated to URLs (recall that we use a weight of 1 if a URL returned for some user query is contained in the ODP).

Another unknown parameter was the amount of output URLs whose associated topics we should investigate / add to the user profile for each query. We experimented with the top 10, 20, 30, 50, 80, and 100 URLs. Optimal results were obtained when exploring the top 30 URLs per query (more URLs usually returned only additional weights, but no more topics, whereas less than 30 URLs sometimes resulted in missing relevant topics).

The complete algorithm for learning user profiles is depicted in Algorithm 5.4.1.1.

Algorithm 5.4.1.1. Learning User Profiles.

1: Let $P[topics, weights]$ be the user profile.
2: **For** each new query q sent to the search engine
3: **For** each output URL u
4: Let $T[topics, 1]$ be the set of topics associated to u.
5: **If** $T \neq \emptyset$
6: $P \leftarrow P \cup T$; **Continue**
7: Let $TR[topics, \lambda]$ be the set of topics associated to $Related(u)$.
8: **If** $TR \neq \emptyset$
9: $P \leftarrow P \cup TR$; **Continue**
10: Let $TH[topics, \mu]$ be the set of topics associated to $HomePage(u)$.
11: **If** $TH \neq \emptyset$
12: $P \leftarrow P \cup TH$; **Continue**
13: **Sort** P decreasingly, after weights.
14: **Return** the top N topics as the user profile.

New topics. User interests have been shown to change over time [151], and our algorithm can clearly keep track of this aspect in the long term. But how to cope with the very first day one searches on a totally different topic? A naïve solution would be to simply temporarily disable the service exploiting the user profile (i.e., personalized search, news, etc.). A more elegant one is to divide the profile into a *permanent profile* PP and a *temporary profile* PT, which itself can also be divided

into the profile built from all today's searches PS and the profile built only from the current search session PC (if any). Then, the profile weights can be computed using the following formula [202]:

$$\begin{aligned} P[t,w] &= 0.6 \cdot PP[t,w] + 0.4 \cdot PT[t,w] = \\ &= 0.6 \cdot PP[t,w] + 0.4 \cdot (0.15 \cdot PS[t,w] + 0.85 \cdot PC[t,w]) \end{aligned}$$

Noise. Even such carefully selected profiles are not perfect, though. Not all previously sent queries are actually relevant to the user's interest: Some of them represent a "one-time" interest, some others relate to a topic no longer addressed by the subject, and so on. However, we argue that these kinds of queries usually represent only a small percentage of the overall search history, and thus their effect over the generated profile is negligible, at least in the long term. Nevertheless, if desired, several methods might be applied in order to filter out noisy queries. For example, one could cluster user's searched expressions by exploiting their Top-K output URLs, and then identify and eliminate the outliers.

5.4.2 Experiments

Experimental Setup

The evaluation of our algorithm was performed using Google Search History[31]. A group of 16 persons created a Google account and used it for a period ranging from 7 to 21 days. After this, they were asked to send us a file containing their queries (pasted from their Google account), which we then used together with the Google API to generate their profiles. Since it was not clear how specific these profiles should be, we investigated the following approaches:

- **Level 2:** Profiles were generated to contain only topics from the second level of the ODP taxonomy or above. More specific topics (i.e., from lower levels) were trimmed / considered as level 2 topics, while the more general topics were left as such.
- **Level 3, Level 4, and Level 5:** Same as above, but with the profiles restricted not to contain topics on levels lower than 3, 4, and 5 respectively.
- **Combined Level 2-4:** Same as "Level 4", but with the weight of each topic multiplied by its depth in the ODP tree. The intuition here is that lower level topics are more specific, and should thus be given more importance.

[31] http://www.google.com/searchhistory/

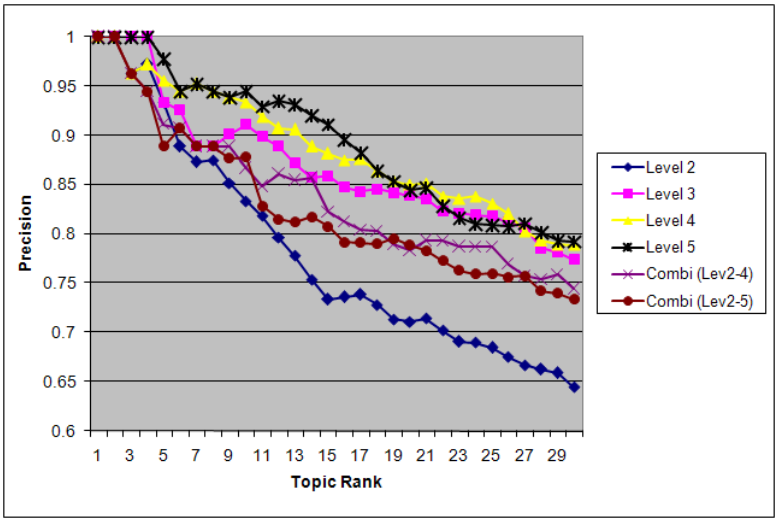

Figure 5.2: Precision at the top 30 topics computed with complete query output.

- **Combined Level 2-5:** Same as "Combined Level 2-4" but down to level 5.

Even though most persons are not interested in more than 10 topics at a time, we generated profiles consisting of the most important 30 topics, in order to fully evaluate the capabilities of our algorithms. Every user had to rate each topic as either irrelevant, or relevant for her interests, the overall results being put together in graphs of *average precision* and *average relative recall*[32] [17], whose input values were averaged over all our testers.

Results

Our results for average precision are depicted in Figure 5.2, and those for average relative recall in Figure 5.3. "Level 5" is outperforming all other algorithms in terms of precision, especially within the Top-20 output topics. Its precision at the Top-10 topics (P@10) is as high as 0.94. At the other end, although it has a slightly better recall than the other algorithms, "Level 2" is clearly not useful for this task because of its quite poor precision. The combined measures yielded only an average performance, indicating that people are not only interested in very specific topics (which received an increased weight with these two metrics), but also in broader, more general topics.

[32]For relative recall, we assumed that the correct results reside within the output $n = 30$ items.

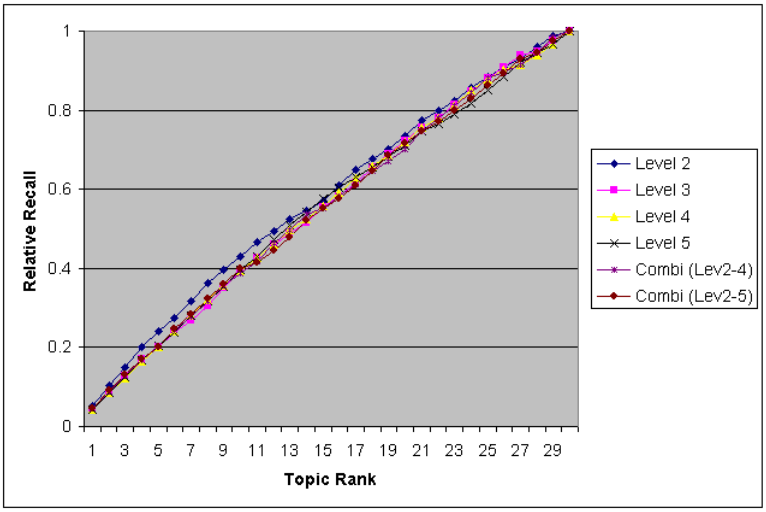

Figure 5.3: Relative Recall at the top 30 topics computed with complete query output.

Further, we wanted to find out how many surfing days were necessary to have the system automatically generate a good enough profile. We experimented with "Level 5" and "Level 2", the latter one because it seemed to perform very well on the best five topics in the previous experiment. As they are quite similar, we only discuss here our results for "Level 5", the better algorithm: Only four days seem to be enough to obtain a precision at 10 above 80% (the average number of URLs clicked per day was 8.75), the more stable output being achieved after six days of using the search engine. The complete average precision results are depicted in Figure 5.4.

5.5 Desktop Based Personalized Web Search

Though very effective, the personalized search algorithm described in Section 5.3.1 still exhibits two drawbacks: First, there is the reduced privacy level. Even though we proposed a less-intrusive profiling technique in Section 5.4, some personal information is nonetheless necessary, i.e., past user queries. Second, there is the limited coverage, as only thouse URLs classified in the ODP are an intrinsic part of the personalization process.

In this section we overcome these last limitations by exploiting user's Personal Information Repository, a self-edited data collection. Several advantages arise when moving Web personalization down to the Desktop level. First, as all "profile"

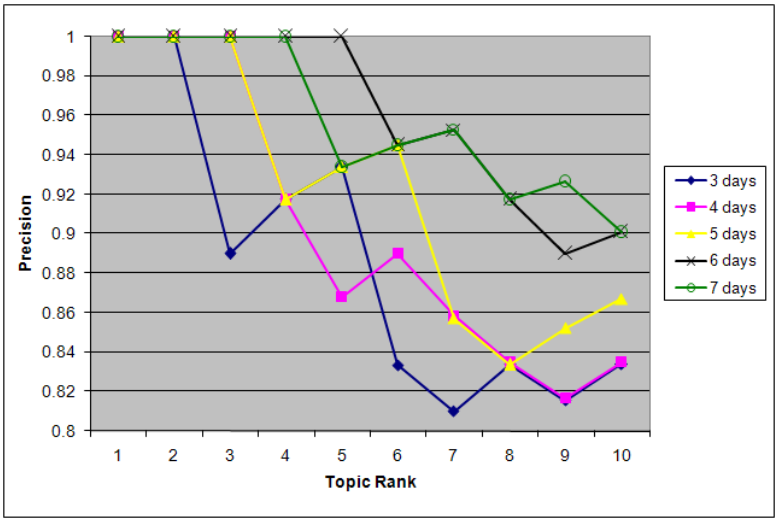

Figure 5.4: Precision per day, using the "Level 5" algorithm.

information is now stored and exploited locally, on the personal machine, complete privacy can be achieved. Search engines should not be able to know about a person's interests, i.e., they should not be able to connect a specific person with the queries she issued, or worse, with the output URLs she clicked within the search interface[33] (see also Volokh [208] for a discussion on the privacy issues related to personalized Web search). Second, this enhanced profile can be used with *any* search algorithm, thus being no longer restricted to ODP. Last, but not least, such a comprehensive description of user interests brings inherently also an increased search quality. After all, the local Desktop is a very rich repository of information, accurately describing most, if not all interests of the user.

The algorithms presented in this section build on top of Web search query expansion. They exploit user's Personal Information Repository to automatically extract additional keywords related both to the query itself and to user's interests, thus implicitly personalizing the search output. The challenge they bring is to provide an efficient utilization of the user profile. Desktop data comes in a very unstructured way, covering documents which are highly diverse in format, content, and even language characteristics. Though projecting this information onto a large taxonomy such as ODP is fairly easy, it is not the optimal solution, as the resulting personalized search algorithm would still suffer from the limited coverage problem. We thus focus on exploiting the PIR as it is, in combination

[33]Generally, search engines can map queries at least to IP addresses, for example by using cookies and mining the query logs. However, by moving the user profile entirely down to the Desktop level we can at least ensure such information is not explicitly associated to a particular user and stored on the search engine side.

with the above mentioned technique of query expansion. We first investigate the analysis of local query context on the Desktop in the first half of Section 5.5.1. We propose several keyword, expression, and summary based techniques for determining relevant expansion terms from those personal documents matching the Web query best. Then, in the second half of Section 5.5.1 we move our analysis to the global Desktop collection and investigate query expansion based on co-occurrence metrics, as well as on filtering a generic thesaurus based algorithm. The experiments presented in Section 5.5.2 show some of these approaches to perform very well, especially on ambiguous queries, producing NDCG [130] improvements of up to 51.28%.

5.5.1 Algorithms

This section presents the five generic approaches for analyzing user's Desktop data in order to provide expansion terms for Web search. In the proposed algorithms we gradually increase the amount of personal information utilized. Thus, in the first part we investigate three local analysis techniques focused only on those Desktop documents matching user's query best. We append to the Web query the most relevant terms, compounds, and sentence summaries from these documents. In the second part of the section we move towards a global Desktop analysis, proposing to investigate term co-occurrences, as well as thesauri, in the expansion process.

Expanding with Local Desktop Analysis

Local Desktop Analysis is related to enhancing Pseudo Relevance Feedback to generate query expansion keywords from the PIR best hits for user's Web query, rather than from the top ranked Web search results. We distinguish three granularity levels for this process and we investigate each of them separately.

Term and Document Frequency. As the simplest possible measures, TF and DF have the advantage of being very fast to compute. Previous experiments with small data sets have showed them to yield very good results [38, 90]. We thus independently associate a score with each term, based on each of the two statistics. The TF based one is obtained by multiplying the actual frequency of a term with a position score descending as the term first appears closer to the end of the document. This is necessary especially for longer documents, because more informative terms tend to appear towards their beginning [38]. The complete TF based keyword extraction formula is as follows:

$$TermScore = \left[\frac{1}{2} + \frac{1}{2} \cdot \frac{nrWords - pos}{nrWords}\right] \cdot \log\left(1 + TF\right) \tag{5.6}$$

where $nrWords$ is the total number of terms in the document and pos is the position of the first appearance of the term; TF represents the frequency of each term in the Desktop document matching user's Web query.

The identification of suitable expansion terms is even simpler when using DF: Given the set of Top-K relevant Desktop documents, generate their snippets as focused on the original search request. This query orientation is necessary, since the DF scores are computed at the level of the entire PIR and would produce too noisy suggestions otherwise. Once the set of candidate terms has been identified, the selection proceeds by ordering them according to the DF scores they are associated with. Ties are resolved using the corresponding TF scores [90].

Note that a hybrid TFxIDF approach is not necessarily efficient, since one Desktop term might have a high DF on the Desktop, while being quite rare in the Web. For example, the term "PageRank" would be quite frequent on the Desktop of an Information Retrieval scientist, thus achieving a low score with TFxIDF. However, as it is rather rare in the Web, it would make a good resolution of the query towards the correct topic.

Lexical Compounds. Anick and Tipirneni [11] 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 several possible compound expressions are available, it has been shown that simple approaches based on noun analysis are almost as good as highly complex part-of-speech pattern identification algorithms [8]. We thus inspect the matching Desktop documents for all their lexical compounds of the following form:

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

All such compounds could be easily generated off-line, at indexing time, for all the documents in the local repository. Moreover, once identified, they can 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: First, the set of relevant Desktop documents is identified; then, a summary containing their most important sentences is generated as output. Sentence selection is the most comprehensive local analysis approach, as it produces the most detailed expansions (i.e., sentences). Its downside is that, unlike with the first two algorithms, its output cannot be stored efficiently, and consequently it cannot be computed off-line. We generate sentence based summaries by ranking

the document sentences according to their salience score, as follows [152]:

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

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:

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

with NS being the total number of sentences in the document (see [152] for details). The second term is a position score set to $(Avg(NS) - SentenceIndex)/Avg^2(NS)$ for the first ten sentences, and to 0 otherwise, $Avg(NS)$ being the average number of sentences over all Desktop items. 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, as longer documents usually do include overall descriptive sentences in the beginning [89], these sentences are more likely to be relevant. The final term biases the summary towards the query. It is the ratio between the square number of query terms present in the sentence and the total number of terms from the query. It is based on the belief that the more query terms contained in a sentence, the more likely will that sentence convey information highly related to the query.

Expanding with Global Desktop Analysis

In contrast to the previously presented approach, global analysis relies on information from across the entire personal Desktop to infer the new relevant query terms. In this section we propose two such techniques, namely term co-occurrence statistics, and filtering the output of an external thesaurus.

Term Co-occurrence Statistics. For each term, we can easily compute offline those terms co-occurring with it most frequently in a given collection (i.e., PIR in our case), and then exploit this information at run-time in order to infer keywords highly correlated with the user query. Our generic co-occurrence based query expansion algorithm is as follows:

Algorithm 5.5.1.1. 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 keywords, potentially similar to an input expression E.
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** $Score(t) \leftarrow \#DesktopHits(E|t)$
6: **Select** Top-K terms of S with the highest scores.

The off-line computation needs an initial trimming phase (step 1) for optimization purposes. In addition, we also restricted the algorithm to computing co-occurrence levels across nouns only, as they contain by far the largest amount of conceptual information, and as this approach reduces the size of the co-occurrence matrix considerably. During the run-time phase, having the terms most correlated with each particular query keyword already identified, one more operation is necessary, namely calculating the correlation of every output term with the entire query. Two approaches are possible: (1) using a product of the correlation between the term and all keywords in the original expression (step 5a), or (2) simply counting the number of documents in which the proposed term co-occurs with the entire user query (step 5b). Small scale tuning experiments performed before the actual empirical analysis indicated the former approach yields a slightly better outcome. Finally, we considered the following Similarity Coefficients [141]:

- *Cosine Similarity*, defined as:

$$CS = \frac{DF_{x,y}}{\sqrt{DF_x \cdot DF_y}} \tag{5.7}$$

- *Mutual Information*, defined as:

$$MI = \log \frac{N \cdot DF_{x,y}}{DF_x \cdot DF_y} \tag{5.8}$$

- *Likelihood Ratio*, defined in the paragraphs below.

DF_x is the Document Frequency of term x, and $DF_{x,y}$ is the number of documents containing both x and y. To further increase the quality of the generated scores we limited the latter indicator to co-occurrences within a window of W terms. We set W to be the same as the maximum amount of expansion keywords desired.

Dunning's Likelihood Ratio λ [87] is a co-occurrence based metric similar to χ^2. It starts by 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 compare the two binomial processes by using likelihood ratios of their associated hypotheses. First, let us define the likelihood ratio for one hypothesis:

$$\lambda = \frac{max_{\omega \in \Omega_0}\ H(\omega; k)}{max_{\omega \in \Omega}\ H(\omega; k)} \tag{5.9}$$

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. If we assume that two binomial distributions have the same underlying parameter, i.e., $\{(p_1, p_2) \mid p_1 = p_2\}$, we can write:

$$\lambda = \frac{max_p\ H(p, p; k_1, k_2, n_1, n_2)}{max_{p_1,p_2}\ H(p_1, p_2; k_1, k_2, n_1, n_2)} \tag{5.10}$$

where $H(p_1, p_2; k_1, k_2, n_1, n_2) = p_1^{k_1} \cdot (1-p_1)^{(n_1-k_1)} \cdot \binom{n_1}{k_1} \cdot p_2^{k_2} \cdot (1-p_2)^{(n_2-k_2)} \cdot \binom{n_2}{k_2}$. Since the maxima are obtained with $p_1 = \frac{k_1}{n_1}$, and $p_2 = \frac{k_2}{n_2}$ for the denominator, and $p = \frac{k_1+k_2}{n_1+n_2}$ for the numerator, we have:

$$\lambda = \frac{max_p\ L(p, k_1, n_1) L(p, k_2, n_2)}{max_{p_1,p_2}\ L(p_1, k_1, n_1) L(p_2, k_2, n_2)} \tag{5.11}$$

where $L(p, k, n) = p^k \cdot (1-p)^{n-k}$. Taking the logarithm of the likelihood, we obtain:

$$\begin{aligned} -2 \cdot \log \lambda = \ & 2 \cdot [\log L(p_1, k_1, n_1) + \log L(p_2, k_2, n_2) - \\ & \log L(p, k_1, n_1) - \log L(p, k_2, n_2)] \end{aligned}$$

where $\log L(p, k, n) = k \cdot \log p + (n-k) \cdot \log(1-p)$. Finally, if we write $O_{11} = P(A\ B)$, $O_{12} = P(\neg A\ B)$, $O_{21} = P(A\ \neg B)$, and $O_{22} = P(\neg A\ \neg B)$, then the

co-occurrence likelihood of terms A and B becomes:

$$\begin{aligned} -2 \cdot \log \lambda = \quad & 2 \cdot [O_{11} \cdot \log p_1 + O_{12} \cdot \log (1 - p_1) + \\ & O_{21} \cdot \log p_2 + O_{22} \cdot \log (1 - p_2) - \\ & (O_{11} + O_{21}) \cdot \log p - (O_{12} + O_{22}) \cdot \log (1 - p)] \end{aligned}$$

where $p_1 = \frac{k_1}{n_1} = \frac{O_{11}}{O_{11}+O_{12}}$, $p_2 = \frac{k_2}{n_2} = \frac{O_{21}}{O_{21}+O_{22}}$, and $p = \frac{k_1+k_2}{n_1+n_2}$.

Thesaurus Based Expansion. Large scale thesauri encapsulate global knowledge about term relationships. Thus, for this technique we first identify the set of terms closely related to each query keyword, and then we calculate the Desktop co-occurrence level of each of these possible expansion terms with the entire initial search request. In the end, those suggestions with the highest frequencies are kept. The algorithm is as follows:

Algorithm 3.1.2.2. Filtered thesaurus based query expansion.

1: **For** each keyword k of an input query Q:
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 PIR with $(Q|t)$, i.e.,
the original query, as expanded with t.
6: **Let** H be the number of hits of the above search
(i.e., the co-occurence level of t with Q).
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 hyponyms (i.e., sub-classes) and meronyms (i.e., sub-parts), and (3) super-concepts, namely hypernyms (i.e., super-classes) and holonyms (i.e., super-parts). As they represent quite different types of association, we investigated them separately. We limited the output expansion set (step 7) to contain only terms appearing at least T times on the Desktop, in order to avoid noisy suggestions, 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 small PIRs.

5.5.2 Experiments

Experimental Setup

We evaluated the quality of our personalization algorithms with 18 subjects (Ph.D. and Post-Doc. students in different areas of computer science and education). First, they installed our Lucene based search engine to index all their locally stored content: Files within user selected paths, Emails, and Web Cache. Without loss of generality, we focused the experiments on single-user machines. Implementing the same algorithms for multi-user computers is a trivial task. Then, our evaluators were asked to choose four queries related to their everyday activities as follows:

- One very frequent AltaVista query, as extracted from the top 2% queries most issued to the search engine within a 7.2 million entries log from October 2001. In order to connect such a query to each user's interests, we added an off-line pre-processing phase. We first generated the most frequent 144,000 AltaVista requests (i.e., 2%), and then randomly selected 50 queries with at least 10 and at most 50 hits on each subject's Desktop. This was necessary in order to ensure that we do not personalize a query which is either not interesting for the user, or of too general interest. In the end, each evaluator had to choose the first of these queries that matched her interests at least partially.
- One randomly selected AltaVista query, filtered using the same procedure as above.
- One self-selected specific query, which they thought to have only one meaning.
- One self-selected ambiguous query, which they thought to have at least three meanings.

The average query lengths were 2.0 and 2.3 terms for the log based queries, as well as 2.9 and 1.8 for the self-selected ones. Even though our algorithms are mainly intended to enhance search when using ambiguous query keywords, we chose to investigate their performance on a wide span of query types, in order to see how they perform in all situations. The log based queries evaluate real life requests[34], in contrast to the self-selected ones, which target rather the identification of top and bottom performances. We should note that the former ones were somewhat farther away from each subject's interest, thus being also more difficult to personalize

[34]Note that at the time when we developed the taxonomy based personalized search algorithm from Section 5.3, we had no real life query log available to test with. Moreover, automatically matching between the query log and user's interests would have been anyway nearly impossible, since no other user specific information is available for that scenario.

on. To gain an insight into the relationship between each query type and user interests, we asked each person to rate the query itself with a score of 1 to 5, having the following interpretations: (1) never heard of it, (2) do not know it, but heard of it, (3) know it partially, (4) know it well, (5) major interest. The obtained grades were 3.11 for the top AltaVista queries, 3.72 for the randomly selected ones, 4.45 for the self-selected specific ones, and 4.39 for the self-selected ambiguous ones. Finally, in order to simplify the experimentation, all Desktop level parts of our algorithms were performed with Lucene using its predefined searching and ranking functions. However, implementing this task using the link analysis ranking algorithms from Chapter 3 would be trivial.

For each of the four test queries, we then collected the Top-5 URLs generated by 20 versions of the algorithms presented in Section 5.5.1. These results were then shuffled into one set containing usually between 70 and 90 URLs. Thus, each subject had to assess about 325 documents for all four queries, being neither aware of the algorithm, nor of the ranking of each assessed URL. Overall, 72 queries were issued and over 6,000 URLs were evaluated during the experiment. For each of these URLs, the testers had to give a rating ranging from 0 to 2, dividing the relevant results in two categories, (1) relevant and (2) highly relevant. Finally, we decided to assess the quality of each ranking using an innovative method, namely the normalized version of Discounted Cumulative Gain (DCG) [130]. DCG is a rich measure, as it gives more weight to highly ranked documents, while also incorporating different relevance levels by giving them different gain values:

$$DCG(i) = \begin{cases} G(1) & , if\ i = 1 \\ DCG(i-1) + G(i)/\log(i) & , otherwise. \end{cases}$$

We used G(i) = 1 for the relevant results, and G(i) = 2 for the highly relevant ones. As queries having more relevant output documents will have a higher DCG, we also normalized its value to a score between 0 (the worst possible DCG given the ratings) and 1 (the best possible DCG given the ratings) to facilitate averaging over queries. All results were tested for statistical significance using T-tests, i.e., we tested whether the improvement over the Google API output[35] is statistically significant.

Algorithmic specific aspects. As our goal is to generate expansion terms for Web queries, it is important to tune the number of such proposed keywords. An initial investigation with a *separate* group of 4 people showed different values to produce the best results across different queries and algorithms. The main influencing factor was by far the query ambiguity level, and we therefore set the

[35]Whenever necessary, we also tested for significance the difference between pairs of the algorithms we proposed.

expansion length to four terms for all proposed techniques, leaving a differentiation at the algorithm level for a future experiment (which we will present later, in Section 5.6).

In order to optimize the run-time computation speed, we chose to limit the number of output keywords per Desktop document to the number of expansion keywords desired (i.e., four). For all algorithms we also investigated bigger limitations. This allowed us to observe that the Lexical Compounds method would perform better if only at most one compound per document were selected. We therefore chose to experiment with this "optimized" approach as well. For all other techniques, considering less than four terms per document did not seem to yield any additional qualitative gain.

In the forthcoming tables, we label the algorithms we evaluated as follows:

0. **Google**: The actual Google query output, as returned by the Google API;
1. **TF, DF**: Term and Document Frequency;
2. **LC, LC[O]**: Regular and Optimized (by considering only one top compound per document) Lexical Compounds;
3. **SS**: Sentence Selection;
4. **TC[CS], TC[MI], TC[LR]**: Term Co-occurrence Statistics using respectively Cosine Similarity, Mutual Information, and Likelihood Ratio as similarity coefficients;
5. **WN[SYN], WN[SUB], WN[SUP]**: WordNet based expansion with synonyms, sub-concepts, and super-concepts, respectively.

Except for the thesaurus based expansion, in all cases we also investigated the performance of our algorithms when exploiting only the Web browser cache to represent user's personal information. This is motivated by the fact that other personal documents such as for example emails are known to have a somewhat different language than that residing on the World Wide Web [204]. We differentiate between these two techniques by adding the $-A$ label suffix to specify that the entire Desktop was used as PIR, and $-W$ where only the Web browser cache was employed.

Results

Log Queries. We evaluated all variants of our algorithms using NDCG. For the case of queries extracted from the search engine log, the best performance was achieved with TF-A, LC[O]-A, and TC[LR]-A. The improvements they brought were up to 5.2% for top queries (with $p = 0.14$) and up to 13.8% for randomly

selected queries (with $p = 0.01$, and thus statistically significant), both being obtained with LC[O]-A. A summary of all results is presented in Table 5.7.

Both TF-A and LC[O]-A yielded very good results, indicating that simple keyword and especially expression oriented approaches might be sufficient for the Desktop based query expansion task. LC[O]-A was much better than LC-A, ameliorating its quality with up to 25.8% in the case of randomly selected log queries, improvement which was also significant with $p = 0.04$. Thus, a selection of compounds spanning over several Desktop documents is more informative about user's interests behind a query than the general approach, in which there is no restriction on the number of compounds produced from every personal item.

The more complex Desktop oriented approaches, namely sentence selection and all term co-occurrence based algorithms, showed a rather average performance, with no visible improvements being noticed (except for TC[LR]-A). Also, the thesaurus based expansion usually produced very few suggestions, possibly because of the many technical queries employed by our subjects. We observed however that expanding with sub-concepts is very good for everyday life terms (e.g., "car"), whereas the use of super-concepts is valuable for compounds having at least one term with low technicality (e.g., "document clustering"). As expected, the synonym based expansion performed generally well, though in some very technical cases it yielded rather general suggestions, which were not filtered out by the Desktop data and thus worsened the query quality.

Two general approaches performed rather poorly. First, DF, even with a snippet based query orientation, still produced quite some frequent and non-resolutive Desktop terms, thus deteriorating retrieval. Second and more important, the use of Web browser cache data to describe user interests, had visibly worse results than Google in all 8 cases. As this was quite unexpected, we interviewed our subjects for clarification. Several reasons have been identified: some of them were using their credit cards quite often for on-line transactions and used to clean their cache frequently in order to protect themselves from malicious intruders; several were running multiple operating systems and used the other one for browsing; others were surfing the Web very rarely, and two of them were recent hires. Also, the Web cache data seemed more noisy, as it also included text from "one time interest" pages, pop-ups, etc. Though we believe these characteristics would less likely show up in other environments, using only the Web browser cache does not seem to be sufficient for extracting good personalized expansion terms for Web search.

Finally, we noticed Google to be very optimized for a set of top frequent queries, making improvements harder for this category of search tasks. Also, even though we investigated quite several approaches to filter out *personalized* queries from

Algorithm	NDCG Top	Signific. vs. Google	NDCG Random	Signific. vs. Google
Google	0.42	-	0.40	-
TF-A	**0.43**	**p = 0.32**	**0.43**	**p = 0.04**
TF-W	0.13	-	0.15	-
DF-A	0.17	-	0.23	-
DF-W	0.12	-	0.10	-
LC-A	0.39	-	0.36	-
LC[O]-A	**0.44**	**p = 0.14**	**0.45**	**p = 0.01**
LC-W	0.13	-	0.13	-
LC[O]-W	0.16	-	0.12	-
SS-A	0.33	-	0.36	-
SS-W	0.11	-	0.19	-
TC[CS]-A	0.37	-	0.35	-
TC[MI]-A	0.40	-	0.36	-
TC[LR]-A	0.41	-	**0.42**	**p = 0.06**
TC[CS]-W	0.19	-	0.14	-
TC[MI]-W	0.18	-	0.16	-
TC[LR]-W	0.17	-	0.17	-
WN[SYN]	0.42	-	0.38	-
WN[SUB]	0.28	-	0.33	-
WN[SUP]	0.26	-	0.26	-

Table 5.7: Normalized Discounted Cumulative Gain at the first 5 results when searching for an AltaVista top (left) and random (right) query.

the search engine log, we should recall that our users were in almost all cases only partially familiar with the topic of the query to evaluate. Thus, our improvements in the range of 10-15% obtained with TF-A and LC[O]-A (both statistically significant) show that such Desktop enhanced query expansion is useful even when searching for queries covering user's interests only marginally.

Self-selected Queries. We also evaluated our algorithms with queries closer to the interests of our subjects. We split them in two categories, clear and ambiguous requests. While our algorithms did not manage to enhance Google for the clear search tasks, they did produce strong improvements of up to 52.9% (which were of course also highly statistically significant with $p \ll 0.01$) when utilized with ambiguous queries. In fact, almost all our algorithms resulted in statistically significant improvements over Google for this query type. A summary of all results is presented in Table 5.8.

Algorithm	**NDCG Clear**	**Signific. vs. Google**	**NDCG Ambiguous**	**Signific. vs. Google**
Google	0.71	-	0.39	-
TF-A	0.66	-	**0.52**	$\mathbf{p \ll 0.01}$
TF-W	0.21	-	0.14	-
DF-A	0.37	-	0.31	-
DF-W	0.17	-	0.11	-
LC-A	0.65	-	**0.54**	$\mathbf{p \ll 0.01}$
LC[O]-A	0.69	-	**0.59**	$\mathbf{p \ll 0.01}$
LC-W	0.16	-	0.15	-
LC[O]-W	0.18	-	0.14	-
SS-A	0.56	-	**0.52**	$\mathbf{p \ll 0.01}$
SS-W	0.20	-	0.11	-
TC[CS]-A	0.60	-	**0.50**	$\mathbf{p = 0.01}$
TC[MI]-A	0.60	-	**0.47**	$\mathbf{p = 0.02}$
TC[LR]-A	0.56	-	**0.47**	$\mathbf{p = 0.03}$
TC[CS]-W	0.17	-	0.14	-
TC[MI]-W	0.13	-	0.13	-
TC[LR]-W	0.22	-	0.18	-
WN[SYN]	0.70	-	0.36	-
WN[SUB]	0.46	-	0.32	-
WN[SUP]	0.51	-	0.29	-

Table 5.8: Normalized Discounted Cumulative Gain at the first 5 results when searching for a user selected clear (left) and ambiguous (right) query.

In general, the relative differences between our algorithms were similar to those observed for the log based queries. As in the previous analysis, the simple Desktop based Term Frequency and Lexical Compounds metrics performed best. Nevertheless, a very good outcome was also obtained for Desktop based sentence selection and all term co-occurrence metrics, indicating that there is still room for Web search improvement in the case of ambiguous queries. There were no visible differences between the behavior of the three different approaches to co-occurrence calculation. Finally, for the case of clear queries, we noticed that fewer expansion terms than four might be less noisy and thus helpful in further improving over the already high quality of the Google output. We thus pursued this idea with the adaptive algorithms presented in the following section.

5.6 Introducing Adaptivity

Both Web search personalization algorithms proposed in the previous sections of this chapter yielded very good results overall. However, neither of them was constant in surpassing the output quality of Google. In particular, the industrial search engine seems to be optimized to handle very common queries, as well as very specific ones. This is why we argue that personalized search algorithms should be flexible, allowing for a combination of regular and user oriented results, both weighted automatically according to the strengths of either approach, to the various aspects of each query, and to the particularities of the person using it. In this section we first discuss the factors influencing the behavior of search algorithms which might be used as input for the adaptivity process. We then show how one of them, namely query clarity, can be applied on top of our Desktop specific query expansion technique. We conclude with an empirical analysis which confirms the additional quality increase brought by the adaptivity feature.

5.6.1 Adaptivity Factors

Several indicators could assist the algorithm to automatically tune the amount of personalization injected into the search output. We will start discussing adaptation by analyzing the query clarity level. Then, we will briefly introduce an approach to model the generic query formulation process in order to additionally tailor the search algorithm automatically, and in the end we will discuss some other possible factors that might be of use for this task.

Query Clarity. The interest for analyzing query difficulty has increased only recently, and there are not many papers addressing this topic. However, it has been long known that query disambiguation algorithms have a high potential of improving retrieval effectiveness for low recall searches with very short queries[36] (see for the example the work of Krovetz and Croft [148]), which is exactly our targeted scenario. Also, the success of Information Retrieval systems clearly varies across different topics (initial work analyzing this phenomenon has been done in the context of the TREC Robust Track [210]). We thus propose to use an estimate number expressing the calculated level of query clarity in order to automatically tweak the amount of personalization fed into the algorithm. The only related work is the paper of Amati et al. [9], who decide on a yes or no basis whether to apply query expansion or not within a search application.

[36]Note that many query disambiguation approaches proposed for other kinds of search scenarios (e.g., collection specific searches with long queries) have not been too successful.

Several approaches have been proposed to quantify query ambiguity. They are as follows:

- *The Query Length* is expressed simply by the number of words in the user query. The solution is not only naïve, but rather inefficient, as reported by He and Ounis [122].
- *The Query Scope* relates to the IDF of the entire query, as in:

$$C_1 = \log\left(\frac{\#DocumentsInCollection}{\#Hits(Query)}\right) \tag{5.12}$$

  This metric performs well when used with document collections covering a single topic, but poor otherwise [77, 122].
- *The Query Clarity* [77] seems to be the best, as well as the most applied technique so far. It measures the divergence between the language model associated to the user query and the language model associated to the collection. In a simplified version (i.e., without smoothing over the terms which are not present in the query), it can be expressed as follows:

$$C_2 = \sum_{w \in Query} P_{ml}(w|Query) \cdot \log \frac{P_{ml}(w|Query)}{P_{coll}(w)} \tag{5.13}$$

  where $P_{ml}(w|Query)$ is the probability of the word w within the submitted query, and $P_{coll}(w)$ is the probability of w within the entire collection of documents.

A number of other solutions exist (see for example [220, 41]), but we think they are too computationally expensive for the huge amount of data that needs to be processed when used on the World Wide Web. We thus decided to investigate the measures C_1 and C_2.

Query Formulation Process. Interactive query expansion has a high potential for enhancing the search experience [186]. We believe that modeling its underlying process would be very helpful in producing qualitative adaptive Web search algorithms. For example, when the user is adding a new term to her previously issued query, she is basically reformulating her original search request. Thus, the newly added terms are more likely to convey information about her underlying search intention. For a general, non personalized retrieval engine, this could correspond to giving more weight to these new query keywords. Within our Desktop based personalized scenario, the generated expansions could similarly be biased towards these terms. Nevertheless, modeling the query reformulation process remains an open challenge. Besides, it is not clear whether regular users are indeed capable of adding topic resolutive terms when they are reformulating their search requests

and it is not straightforward to accurately separate real query reformulations from simple changes in user's search goal.

Other Features. The general idea of adapting the retrieval process to various aspects of the query, of the user himself, and even of the employed algorithm has received only little attention in the scientific literature. Only some approaches have been investigated, usually in an indirect way. There exist studies of several indicators, such as query behaviors at different times of day, or of the topics spanned by the queries of various classes of users, or of the preponderant locations and languages employed for searching the Web, etc. (e.g., [24]). However, they generally do not discuss how these features can be actually incorporated in the search process itself. Moreover, they have almost never been related to the task of Web personalization, even though they are more important for this latter case, as the personalized search algorithms should be flexible and able to optimally configure themselves for each search request in an automatic way.

5.6.2 Desktop Based Adaptive Personalized Search

As the query clarity indicators were the only ones sufficiently developed in order to be used within search algorithms, we decided to integrate them into our Desktop specific query expansion technique, so as to evaluate the feasibility of using an adaptive personalized search system. We settled onto building upon C_1 and C_2, as discussed in the previous section, and started by analyzing their performance over a large set of queries. The resulting clarity predictions were split into three categories:

- Small Scope / Clear Query: $C_1 \in [0, 12], C_2 \in [4, \infty)$.
- Medium Scope / Semi-Ambiguous Query: $C_1 \in [12, 17), C_2 \in [2.5, 4)$.
- Large Scope / Ambiguous Query: $C_1 \in [17, \infty), C_2 \in [0, 2.5]$.

Surprisingly, the same intervals were quite well delimited on both the Web and personal repositories and the two measures C_1 and C_2 produced rather similar predictions. In order to limit the size of the experimental process, we chose to analyze only the results produced when employing C_1 for the PIR and C_2 for the Web, as this was the best combination (by a small margin), as observed from an initial manual analysis of the output produced by each metric.

As an algorithmic basis we employed LC[O]-A, i.e., personalized query expansion using optimized lexical compounds within the entire user PIR, as it was clearly the winning method in the previous analysis. However, an investigation of the expansion terms it generated showed it to slightly overfit the results for clear queries. We therefore utilized a substitute for this particular case. Two candidates

Desktop Scope	Web Clarity	No. of Terms	Algorithm
Large	Ambiguous	4	LC[O]-A
Large	Semi-Ambig.	3	LC[O]-A
Large	Clear	2	LC[O]-A
Medium	Ambiguous	3	LC[O]-A
Medium	Semi-Ambig.	2	LC[O]-A
Medium	Clear	1	TF-A / WN[SYN]
Small	Ambiguous	2	TF-A / WN[SYN]
Small	Semi-Ambig.	1	TF-A / WN[SYN]
Small	Clear	0	-

Table 5.9: Adaptive Personalized Query Expansion.

were considered: (1) TF-A, i.e., term frequency with all Desktop data, as it was the second best approach overall, and (2) WN[SYN], i.e., WordNet synonyms, as we observed that its first and second expansion terms were often very good.

Given the algorithms and the clarity measures, we implemented the adaptivity procedure by tailoring the amount of expansion terms added to the original query, as a function of its ambiguity in the Web, as well as within the Personal Information Repository of the user. Note that the ambiguity level is related to the number of documents covering a certain query. Thus, to some extent, it has different meanings on the Web and within PIRs. While a query deemed ambiguous on a large collection such as the Web will very likely indeed have a large number of meanings, this may not be the case for the Desktop. Take for example the query "PageRank". If the user is a link analysis expert, many of her documents might match this term, and thus the query would be classified as ambiguous. However, when analyzed against the Web, this is definitely a clear query. Consequently, we employed more additional query terms when the query was more ambiguous in the Web, but also on the Desktop. Put another way, queries deemed clear on the Desktop were inherently not well covered within user's Personal Information Repository, and thus had fewer keywords appended to them. The actual number of expansion terms we utilized for each combination of scope and clarity levels is depicted in Table 5.9.

Note that we also requested to have at least 20 hits of the original query on the local Desktop, in order to cope with too shallow PIRs, either because the machine was new, or because the topic did not represent a common interest of the user. Whenever this constraint was not satisfied, the query was simply left unexpanded.

5.6.3 Experiments

Experimental Setup

For the empirical analysis of our adaptive search algorithms we used exactly the same experimental setup as for our previous Desktop personalized query expansion techniques, with two log-based queries and two self-selected ones (all different from before), evaluated with NDCG over the Top-5 results output by each algorithm.

The approaches included into this final study were as follows:

0. **Google**: The actual Google query output, as returned by the Google API;
1. **TF-A**: Term Frequency, over the entire PIR;
2. **LC[O]-A**: Optimized (by considering only one top compound per document) Lexical Compounds, also using all available user data;
3. **WN[SYN]**: WordNet based expansion with synonyms;
4. **A[LCO/TF]**: Adaptive personalized search with TF-A for clear Desktop queries, and LC[O]-A otherwise;
5. **A[LCO/WN]**: Same as above, but with WN[SYN] used instead of TF-A.

Results

The overall results were at least similar, or better than Google for all kinds of log queries (see Table 5.10). In the case of top frequent queries, both adaptive algorithms, A[LCO/TF] and A[LCO/WN], improve Google with 10.8% and 7.9% respectively, both differences being also statistically significant with $p \leq 0.01$. They also achieve an improvement of up to 6.62% over the best performing static algorithm, LC[O]-A (the p-value in this case being 0.07). For randomly selected queries, even though A[LCO/TF] manages to yield significantly better results than Google ($p = 0.04$), both adaptive approaches fall behind the static algorithms. The major reason for this seems to be the unstable dependency of the number of expansion terms, as a function of query clarity.

The analysis of the self-selected queries shows that adaptivity can bring even further improvements into the Web search personalization process (see Table 5.11). For ambiguous queries, the scores given to Google search are enhanced by 40.6% through A[LCO/TF] and by 35.2% through A[LCO/WN], both strongly significant with $p \ll 0.01$. Allowing for flexibility in the number of expansion keywords brings another 8.9% improvement over the static personalization of LC[O]-A ($p = 0.05$). Even in the case of clear queries, the adaptive algorithms perform better (though only with a very small margin), improving over Google with 0.4% and 1.0% respectively (see Figure 5.5).

Algorithm	NDCG Top	Signific. vs. Google	NDCG Random	Signific. vs. Google
Google	0.51	-	0.45	-
TF-A	0.51	-	**0.48**	$\mathbf{p = 0.04}$
LC[O]-A	**0.53**	$\mathbf{p = 0.09}$	**0.52**	$\mathbf{p < 0.01}$
WN[SYN]	0.51	-	0.45	-
A[LCO/TF]	**0.56**	$\mathbf{p < 0.01}$	**0.49**	$\mathbf{p = 0.04}$
A[LCO/WN]	**0.55**	$\mathbf{p = 0.01}$	0.44	-

Table 5.10: Normalized Discounted Cumulative Gain at the first 5 results when using our *adaptive* personalized search algorithms on an AltaVista top (left) and random (right) query.

Algorithm	NDCG Clear	Signific. vs. Google	NDCG Ambiguous	Signific. vs. Google
Google	0.81	-	0.46	-
TF-A	0.76	-	**0.54**	$\mathbf{p = 0.03}$
LC[O]-A	0.77	-	**0.59**	$\mathbf{p \ll 0.01}$
WN[SYN]	0.79	-	0.44	-
A[LCO/TF]	**0.81**	-	**0.64**	$\mathbf{p \ll 0.01}$
A[LCO/WN]	**0.81**	-	**0.63**	$\mathbf{p \ll 0.01}$

Table 5.11: Normalized Discounted Cumulative Gain at the first 5 results when using our *adaptive* personalized search algorithms on a user selected clear (left) and ambiguous (right) query.

All results are depicted graphically in Figure 5.5. We notice that A[LCO/TF] is the overall best algorithm, performing better than Google for all types of queries, either extracted from the search engine log, or self-selected. Therefore, the experiments presented in this section confirm clearly that adaptivity is a necessary further step to take.

5.7 Discussion

The billions of pages available on the World Wide Web, together with the continuous attempts of spammers to artificially promote low quality content, have determined information finding to become a more and more difficult task in this environment. It is only the extensive work performed within the search engine industry that kept search quality at reasonably good levels. Personalization comes

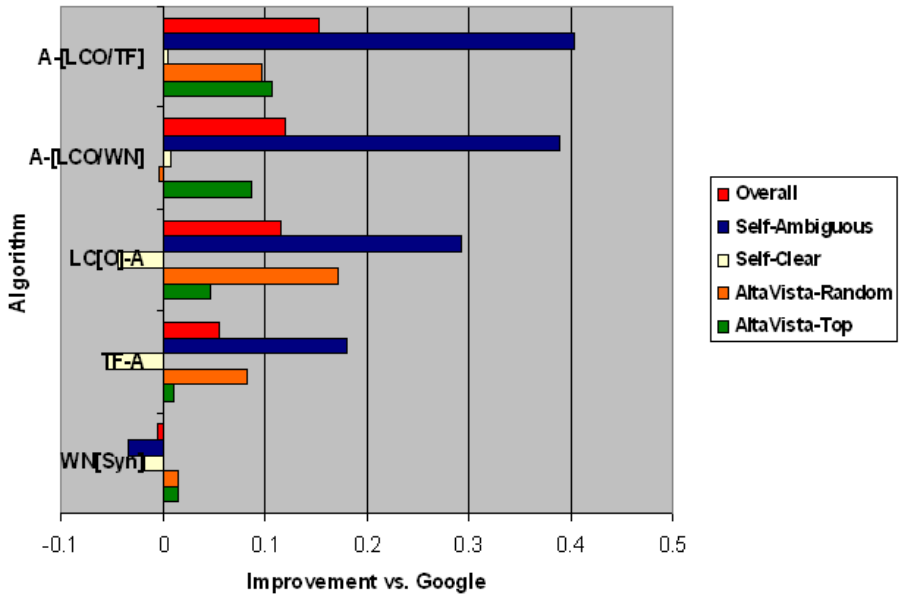

Figure 5.5: Relative NDCG gain (in %) for each algorithm overall, as well as separated per query category.

to back this effort by tailoring the search results according to each user's interests, thus implicitly bringing more order into the output and demoting spam pages.

In this chapter we proposed a series of algorithms meant to overcome the limitations of current Web search personalization approaches. We first showed how to generalize personalized search in catalogues such as ODP and Google Directory beyond the currently available search restricted to specific categories, yielding simple yet very effective personalization algorithms. The precision achieved by our approach significantly surpassed the precision offered by Google in several sets of extensive experiments. The big plus of this initial algorithm is its very fast, straightforward computation, backed by the very good output quality. The minuses are its lack of automatic user profiling and its reduced coverage, which is limited to the contents of the input taxonomy.

We then tackled the first challenge brought by our ODP enabled approach and described a new algorithm that automatically learns user profiles in online systems, based on an analysis of search statistics. This was either faster, or less obtrusive than any previous approach. Nevertheless, it still required some small amount of user specific information.

In the third part of the chapter we introduced an algorithm meant to overcome all problems posed by the previously existing approaches, including the above men-

tioned ones. More specifically, we proposed five techniques for determining query expansion terms by analyzing the Personal Information Repository of each user. Each of them produced additional keywords by mining Desktop data at increasing granularity levels, ranging from term and expression level analysis up to global co-occurrence statistics and external thesauri. Just as before, we provided a thorough empirical analysis of several variants of our approaches, under four different scenarios. Some of these were showed to perform very well, especially on ambiguous queries. All in all, this last algorithm had no privacy implications, complete Web coverage, and automatic profiling with no user interaction necessary, all at the cost of a minimal additional overhead put onto the search process.

To make our results even better we proposed to make the entire personalization process adaptive to the features of each query, a strong focus being put on clarity level. Within another separate set of experiments, we showed these adaptive algorithms to provide further improvements over our previously identified best approach. In fact, with adaptivity in place, our technique was better than Google under all tested search scenarios.

A series of fine tuning studies might still be interesting to add. For example, there is no investigation upon the dependency between various query features and the optimal number of expansion terms. Similarly, there is still no technique to define the best weighting of terms within the query, either within regular search, or especially within personalized algorithms. Last, but not least, new adaptivity metrics based for example on the query reformulation patterns would most probably allow for additional quality increases.

Chapter 6

Conclusions and Open Directions

As data storage capacities become larger every day, the need for effective information organization algorithms grows as well. In this thesis we identified and analyzed in detail those applications with the most imperative demand for such a structuring, namely Desktop Search, Spam Detection, and Web Search. We proposed link analysis ranking as a solution for enhancing data access, building upon the underlying social characteristics of each application context, either at the macroscopic, or the microscopic level. This section first summarizes our major research contributions with respect to the three above mentioned domains, and then discusses some issues which remained open for future investigations.

Summary of Contributions

The main focus of our work was to deploy link analysis ranking solutions for information management within all areas in which practical data organization approaches were either completely inexistent, or only partially developed.

We started in Chapter 2 with a comprehensive overview of link analysis ranking algorithms, as it represents the foundation of our approaches. First, we placed the ranking module into a generic search engine architecture, thus showing how our techniques could be deployed into practical applications. Then, we presented a detailed discussion of all major aspects of PageRank, the supporting link analysis algorithm used within our approaches. The main aspects we covered included convergence, stability, treatment dangling nodes, implementation and optimization mechanisms, as well as possible future extensions. Consequently, we presented

a similar overview for HITS, another important milestone algorithm within the history of link analysis. In the final part of the chapter we introduced the reader to some other non-link features used for ranking Web pages. Also, before moving to the core of the thesis, we pin pointed the other applications in need of data organization mechanisms and motivated the selection of the above mentioned three ones.

In Chapter 3 we argued that all previous search based solutions to locate information at the PC Desktop level are insufficient for the scalability requirements imposed by current storage devices. We therefore introduced a totally new Desktop Ranking technique which builds upon link analysis in order to rank items within Personal Information Repositories. Two approaches were taken when implementing the idea. First, we exploited contextual analysis of specific user actions in order to build a link structure over the PIR and to compute "local reputation" values for personal items. Second, we generalized the algorithm by including all user actions into the ranking algorithm, solution which was clearly much simpler, while also providing a visibly larger coverage of the personal items. Both techniques were thoroughly investigated empirically, and in both cases the Desktop search output quality was strongly increased, improving over the regular TFxIDF ranking used by current applications at a statistically significant difference.

Chapter 4 discussed Spam Detection as a bridging application between personal Desktops, social networks and the Internet. We tackled the two forms of spam which are by far most spread: Email spam and Web spam. First, we proposed MailRank, a link analysis based approach to Email ranking and classification which intelligently exploits the social communication network created via Email exchanges. The algorithm collects its input values from the sent-mail folder of all participants, and then applies a power-iteration technique in order to rank trustworthy senders and to detect spammers. An experimental analysis showed MailRank to yield very accurate anti-spam decisions, stable in the presence of sparse networks or of various malicious attacks, while also bringing an additional, new research benefit: Ordering personal Emails according to the social reputation of their sender. In the second part of the chapter we applied the very same background to design a novel approach to remove artificial patterns from Web hyperlink structures. We proposed to utilize link analysis at the level of Web sites, in contrast to the current approaches, built at the page level. We designed and evaluated algorithms tackling three types of inappropriate site level relationships: (1) Mutual reinforcements between Web sites, (2) Abnormal support from one site towards another, and (3) Link alliances across multiple sites. Our experiments on top of the link database of the TodoBR search engine showed a quality improvement of up to about 60% in Mean Average Precision.

Our most important contributions are those tackling the Web search application, described in Chapter 5. Even though an extensive Web ranking research does exist already, the search output quality is still average. The chapter proposed a series of algorithms meant to overcome existing Web search limitations through enhanced user personalization. First, we showed how to provide fast personalization based on large scale Web taxonomies by introducing an additional criterion for Web page ranking, namely the (link) distance between a user profile defined with taxonomical themes and the sets of topics covered by each URL returned in Web search. The precision achieved by this technique significantly surpassed the precision offered by Google search, reaching up to 63% in quality improvement. We then described a new algorithm that automatically learns user profiles not only for our taxonomy based approach, but for any online system, exploiting simple statistics built on top of the taxonomy topics associated to the output of each user's Web queries. The precision of this mechanism was 94% when computed at the Top-10 user interests, a value above 80% being reached after only 4 days of regular Web searching. The second half of the chapter introduced a further enhanced personalized search framework which brought three major additional improvements over our taxonomy based approach: (1) No privacy implications, (2) Complete Web coverage unrestricted to the size of the taxonomy, and (3) Automatic profiling with no user interaction necessary. More specifically, we proposed five techniques for determining Web query expansion terms by analyzing the Personal Information Repository of each user. Each of them produced additional keywords by mining Desktop data at increasing granularity levels, ranging from term and expression level analysis up to global co-occurrence statistics and external thesauri. Some of these techniques performed very well, especially on queries deemed ambiguous by our testers. Finally, to bring our results even closer to the user needs, we investigated the design of a search process adaptive to the features of each query, especially to its clarity level. This last approach provided additional improvements over all our previous algorithms and yielded better quality output than Google under any tested scenario.

The headline results of this thesis are: (1) The initiation of a new Desktop research stream oriented on Usage Analysis (also considered by TREC[1] for addition as main track in 2008), (2) An increased awareness of the importance of both Desktop data and large scale taxonomies for Web Search Personalization, (3) The general idea of exploiting social network information for Email spam detection, as well as (4) Quite several algorithm and system design considerations for link analysis enabled applications.

[1]Text REtrieval Conference, http://trec.nist.gov.

Open Directions

Research on a topic is almost never *complete*. New ideas generate new problems, which in turn generate new ideas, and so on. Especially as some of our contributions opened new research paths, there exist a few areas we find interesting for future investigation. Let us discuss them separated onto the applications we tackled in the thesis.

Desktop Search. This is the most rife domain with respect to possible additional steps. First, it remains unclear how to accurately detect personal items which are accessed together *and* also belong to the same activity context. Many approaches could be envisioned, for example by automatically trimming the Desktop links generated by our current generalized technique, or by developing enhanced linking heuristics, or simply by developing an improved ranking algorithm. Second, one might also investigate other ranking extensions which include non access based heuristics as well, thus addressing more local resources, even when these have never been opened by the user. Third, since Desktop Usage Analysis has not been investigated in the past, it could thus be exploited for quite a lot of other ideas. For instance one could attempt to combine textual evidences with usage analysis in order to achieve a more exact clustering or context detection over Personal Information Repositories, or just over Email inboxes, and so on.

Spam Detection. Though being very accurate, our Email reputation approach to spam detection has not entirely solved the problem. If broadly accepted by the community, link analysis could be further investigated to provide more complex, secure algorithms, optimized for neutralizing malicious users, in a similar fashion to the Web approaches. Also, as wide usage would imply less scalability, MailRank or its enhanced follow-up Email anti-spam algorithm should be moved towards a distributed infrastructure, thus implicitly allowing for more possible points of failure as well. In general, we believe that the power of social networks has not yet been fully exploited within the current spam detection systems, and thus much more is yet to come. With respect to Web spam detection, there are two broad problems which we left unsolved. First, it might be beneficial to assess the "spam level" of each link, and weight the links accordingly, rather than removing the malicious candidates entirely. Second, in a more general sense, it is still unclear how to optimally combine the already several complementary Web spam detection approaches, such as the page level and the site level algorithms. Quite several solutions are possible, for example to apply them in a sequence, or altogether, weighted according to the predictions of some machine learning algorithm, etc.

Web Search Personalization. While we strived to bring Web Search Personalization as close to perfection as possible, there are of course some issues in need

of further investigation. Our Desktop based query expansion framework would clearly benefit from an investigation upon the dependency between various query features and the optimal number of expansion terms. Similarly, there is still no technique to define the best weighting of terms within the query, either within regular search, or especially within personalized algorithms. Last, and most challenging, we believe it would be highly beneficial to model the query reformulation process of each user. Providing such an approach to learn search behaviors would in the future allow to "personalize the personalization algorithm", thus bringing an ultimate adaptivity into search. Yet many unknowns exist there: Which are the metrics that best indicate user's behavior? Which is the importance of each of them? How should they be combined?

We still find ourselves in an early phase of the digital information era. This is why we believe a lot of ranking research is still to come, either built on top of link or text analysis, or on top of more complex statistical metrics, etc. Besides the already existing wide interest for ranking in the Web, a lot more collateral Information Management problems will appear, such as information organization and alignment within a company Intranet, information structuring within specialized repositories (e.g., a server dedicated to company reports), and so on. For all of them, ranking will help, either by playing a minor or even a major role in the ultimate solution to each problem.

Bibliography

[1] K. Aberer and J. Wu. A framework for decentralized ranking in web information retrieval. In *The Fifth Asia Pacific Web Conference, APWeb*, 2003.

[2] S. Abiteboul, M. Preda, and G. Cobena. Adaptive on-line page importance computation. In *Proc. of the 12th Intl. Conf. on World Wide Web*, 2003.

[3] S. Acharyya and J. Ghosh. Outlink estimation for pagerank computation under missing data. In *Proc. of the 13th Intl. World Wide Web Conf.*, 2004.

[4] E. Adar, D. Kargar, and L. A. Stein. Haystack: per-user information environments. In *Proc. of the 8th Intl. CIKM Conf. on Information and Knowledge Management*, 1999.

[5] M. S. Aktas, M. A. Nacar, and F. Menczer. Personalizing pagerank based on domain profiles. In *Proc. of the KDD Workshop on Web Mining and Usage Analysis held in conjunction with the 10th ACM International SIGKDD Conference*, 2004.

[6] R. Albert, H. Jeong, and A.-L. Barabási. Diameter of the World Wide Web. *Nature*, 401:130–131, September 1999.

[7] B. Aleman-Meza, C. Halaschek, I. B. Arpinar, and A. Sheth. Context-aware semantic association ranking. In *Semantic Web and Databases Workshop*, 2003.

[8] J. Allan and H. Raghavan. Using part-of-speech patterns to reduce query ambiguity. In *Proc. of the 25th Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2002.

[9] G. Amati, C. Carpineto, and G. Romano. Query difficulty, robustness, and selective application of query expansion. In *Proc. of the ECIR European Conf. on Information Retrieval*, 2004.

[10] E. Amitay, D. Carmel, A. Darlow, R. Lempel, and A. Soffer. The connectivity sonar: detecting site functionality by structural patterns. In *Proceedings of the 14th ACM Conference on Hypertext and Hypermedia*, 2003.

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

[12] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin. Pagerank computation and the structure of the web: Experiments and algorithms. In *Proc. of the 11th Intl. World Wide Web Conf.*, 2002.

[13] G. Attardi, A. Gullí, and F. Sebastiani. Automatic web page categorization by link and context analysis. In *Proc. of the THAI European Symp. on Telematics, Hypermedia and Artificial Intelligence*, 1999.

[14] R. Baeza-Yates, P. Boldi, and C. Castillo. Generalizing pagerank: Damping functions for link-based ranking algorithms. In *Proc. of the 29th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2006.

[15] R. Baeza-Yates, C. Castillo, and V. López. Pagerank increase under different collusion topologies. In *First International Workshop on Adversarial Information Retrieval on the Web*, 2005.

[16] R. Baeza-Yates and E. Davis. Web page ranking using link attributes. In *Proc. of the 13th Intl. World Wide Web Conf.*, 2004.

[17] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[18] A. Balmin, V. Hristidis, and Y. Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *Proc. of the Intl. VLDB Conf. on Very Large Databases*, 2004.

[19] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

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

[21] L. Becchetti and C. Castillo. The distribution of pagerank follows a power-law only for particular values of the damping factor. In *Proc. of the 15th Intl. Conf. on World Wide Web*, 2006.

[22] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Link-based characterization and detection of web spam. In *Prod. of the 2nd Intl. AIRWeb Workshop on Adversarial Information Retrieval on the Web*, 2006.

[23] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Using rank propagation and probabilistic counting for link-based spam detection. In *Proc. of the WebKDD Workshop on Web Mining and Web Usage Analysis*, 2006.

[24] S. M. Beitzel, E. C. Jensen, A. Chowdhury, D. Grossman, and O. Frieder. Hourly analysis of a very large topically categorized web query log. In *Proc. of the 27th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2004.

[25] A. A. Benczur, K. Csalogany, T. Sarlos, and M. Uher. Spamrank - fully automatic link spam detection. In *First International Workshop on Adversarial Information Retrieval on the Web*, 2005.

[26] K. Bharat, A. Z. Broder, J. Dean, and M. R. Henzinger. A comparison of techniques to find mirrored hosts on the WWW. *Journal of the American Society of Information Science*, 51(12):1114–1122, 2000.

[27] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proc. of the 21st Intl. SIGIR Conf. on Research and Development in Information Retrieval*, 1998.

[28] A. Bifet, C. Castillo, P. A. Chirita, and I. Weber. An analysis of factors used in search engine ranking. In *Proc. of the 1st Workshop on Adversarial Information Retrieval held at the 14th Intl. World Wide Web Conf.*, 2005.

[29] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas. Finding authorities and hubs from link structures on the world wide web. In *Proc. of the 10th Intl. Conf. on World Wide Web*, 2001.

[30] J. Bortz. *Statistics for Social Scientists*. Springer, 1993.

[31] P. Boykin and V. Roychowdhury. Leveraging social networks to fight spam. *IEEE Computer*, 38(4):61–68, 2005.

[32] S. Brin, R. Motwani, L. Page, and T. Winograd. What can you do with a web in your pocket? *Data Engineering Bulletin*, 21(2):37–47, 1998.

[33] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[34] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. In *Proc. of the 9th Intl. World Wide Web Conf.*, 2000.

[35] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Comput. Netw. ISDN Syst.*, 29(8-13):1157–1166, 1997.

[36] A. Z. Broder, R. Lempel, F. Maghoul, and J. Pedersen. Efficient pagerank approximation via graph aggregation. In *Proc. of the 13th Intl. World Wide Web Conf.*, 2004.

[37] 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 *Proc. of the Semantic Desktop Workshop held at the 5th Intl. Semantic Web Conf.*, 2006.

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

[39] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proc. 22nd International Conf. on Machine Learning*, 2005.

[40] D. Carmel, E. Farchi, Y. Petruschka, and A. Soffer. Automatic query wefinement using lexical affinities with maximal information gain. In *Proc. of the 25th Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2002.

[41] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? In *Proc. of the 29th Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2006.

[42] C. Carpineto, R. de Mori, G. Romano, and B. Bigi. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems*, 19(1):1–27, 2001.

[43] J. Carriere and R. Kazman. Webquery: Searching and visualizing the Web through connectivity. In *Proceedings of the 6th International World Wide Web Conference*, 1997.

[44] A. Carvalho, P. A. Chirita, E. S. de Moura, P. Calado, and W. Nejdl. Site level noise removal for search engines. In *Proc. of the 15th Intl. World Wide Web Conf.*, 2006.

[45] S. Chakrabarti. Integrating the document object model with hyperlinks for enhanced topic distillation and information extraction. In *Proc. of the 10th Intl. Conf. on World Wide Web*, 2001.

[46] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, 2003.

[47] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference*, 1998.

[48] P. Chan. Constructing web user profiles: A non-invasive learning approach. *Web Usage Analysis and User Profiling, Springer LNAI 1836*, 2000.

[49] C.-H. Chang and C.-C. 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.

[50] L. Chen and P. Pu. Survey of preference elicitation methods. Technical report, IC/2004/67 - EPFL, 2004.

[51] Y.-Y. Chen, Q. Gan, and T. Suel. I/o-efficient techniques for computing pagerank, 2002.

[52] S. Chernov, P. Serdyukov, P.-A. Chirita, G. Demartini, and W. Nejdl. 2. building a desktop search test-bed. In *Proc. of the 29th European Conference on Information Retrieval*, 2006.

[53] A. Cheyer, J. Park, and R. Giuli. Iris: Integrate. relate. infer. share. In *Proc. of the 1st Workshop on the Semantic Desktopheld at the Intl. Semantic Web Conf.*, 2005.

[54] S. Chien, C. Dwork, R. Kumar, D. Simon, and D. Sivakumar. Link evolution: Analysis and algorithms. *Internet Mathematics*, 1(3):277–304, 2004.

[55] P. A. Chirita, S. Costache, J. Gaugaz, and W. Nejdl. Desktop context detection using implicit feedback. In *Proc. of the Personal Information Management Workshop held at the 29th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2006.

[56] P. A. Chirita, S. Costache, S. Handschuh, and W. Nejdl. P-tag: Large scale automatic generation of personalized annotation tags for the web. In *Proc. of the 16th Intl. World Wide Web Conf.*, 2007.

[57] P. A. Chirita, A. Damian, W. Nejdl, and W. Siberski. Search strategies for scientific collaboration networks. In *Proc. of the P2P Information Retrieval Workshop held at the 14th ACM Intl. CIKM Conf. on Information and Knowledge Management*, 2005.

[58] P. A. Chirita, J. Diederich, and W. Nejdl. Mailrank: Using ranking for spam detection. In *Proc. of the 14th Intl. CIKM Conf. on Information and Knowledge Management*, 2005.

[59] P. A. Chirita, C. S. Firan, and W. 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 Conf. on Information and Knowledge Management*, 2006.

[60] P. A. Chirita, C. S. Firan, and W. Nejdl. Summarizing local context to personalize global web search. In *Proc. of the 15th Intl. CIKM Conf. on Information and Knowledge Management*, 2006.

[61] P. A. Chirita, R. Gavriloaie, S. Ghita, W. Nejdl, and R. Paiu. Activity based metadata for semantic desktop search. In *Proc. of the 2nd European Semantic Web Conference*, Heraklion, Greece, 2005.

[62] P. A. Chirita, S. Ghita, W. Nejdl, and R. Paiu. Semantically enhanced searching and ranking on the desktop. In *Proc. of the Semantic Desktop Workshop held at the 4th Intl. Semantic Web Conf.*, 2005.

[63] P. A. Chirita, S. Ghita, W. Nejdl, and R. Paiu. Beagle++: Semantically enhanced searching and ranking on the desktop. In *Proc. of the 3rd European Semantic Web Conference*, 2006.

[64] P. A. Chirita, S. Idreos, M. Koubarakis, and W. Nejdl. Designing semantic publish/subscribe networks using super-peers. In *Semantic Web and Peer-To-Peer (book)*, Jan 2004.

[65] P. A. Chirita, S. Idreos, M. Koubarakis, and W. Nejdl. Publish/subscribe for RDF-based P2P networks. In *Proc. of the 3rd European Semantic Web Conference*, 2004.

[66] P. A. Chirita and W. Nejdl. Analyzing user behavior to rank desktop items. In *Proc. of the 13th Intl. Symp. on String Processing and Information Retrieval (SPIRE)*, 2006.

[67] P. A. Chirita, W. Nejdl, R. Paiu, and C. Kohlschütter. Using odp metadata to personalize search. In *Proc. of the 28th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2005.

[68] P. A. Chirita, W. Nejdl, M. Schlosser, and O. Scurtu. Personalized reputation management in p2p networks. In *Proc. of the Workshop on Trust, Security and Reputation held at the 3rd Intl. Semantic Web Conf.*, 2004.

[69] P. A. Chirita, W. Nejdl, and O. Scurtu. Knowing where to search: Personalized search strategies for peers in p2p networks. In *Proc. of the P2P Information Retrieval Workshop held at the 27th Intl. ACM SIGIR Conf.*, 2004.

[70] P. A. Chirita, W. Nejdl, and C. Zamfir. Preventing shilling attacks in online recommender systems. In *Proc. of the 7th ACM Intl. Workshop on Web Information and Data Management held at the 14th Intl. ACM CIKM Conf. on Information and Knowledge Management*, 2006.

[71] P. A. Chirita, D. Olmedilla, and W. Nejdl. Finding related hubs and authorities. In IEEE, editor, *In Proc. of the First Latin American Web Congress (LA-WEB)*, Santiago, Chile, Nov 2003.

[72] P. A. Chirita, D. Olmedilla, and W. Nejdl. Finding related pages on the link structure of the www. In *Proc. of the 3rd IEEE/WIC/ACM Intl. Conf. on Web Intelligence*, 2004.

[73] P. A. Chirita, D. Olmedilla, and W. Nejdl. Pros: A personalized ranking platform for web search. In *Proc. of the 3rd Intl. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2004.

[74] M. Claypool, D. Brown, P. Le, and M. Waseda. Inferring user interest. *IEEE Internet Computing*, 5(6), 2001.

[75] D. Cohn and H. Chang. Learning to probabilistically identify authoritative documents. In *Proc. 17th International Conf. on Machine Learning*, 2000.

[76] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.

[77] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proc. of the 25th Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2002.

[78] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *Proc. of the 11th Intl. Conf. on World Wide Web*, 2002.

[79] B. Davison. Recognizing nepotistic links on the web. In *Proceedings of the AAAI-2000 Workshop on Artificial Intelligence for Web Search*, 2000.

[80] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[81] J.-Y. Delort, B. Bouchon-Meunier, and M. Rifqi. Enhanced web document summarization using hyperlinks. In *Proc. of the 14th ACM HYPERTEXT Conf. on Hypertext and Hypermedia*, 2003.

[82] M. Diligenti, M. Gori, and M. Maggini. A unified probabilistic framework for web page scoring systems. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):4–16, 2004.

[83] C. Ding, X. He, P. Husbands, H. Zha, and H. D. Simon. Pagerank, hits and a unified framework for link analysis. In *Proc. of the 25th Intl. SIGIR Conf. on Research and Development in Information Retrieval*, 2002.

[84] 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 CIKM Conf. on Information and Knowledge Management*, 2004.

[85] X. L. Dong and A. Halevy. A platform for personal information management and integration. In *Proc. of the 2nd Conf. on Innovative Data Systems Research (CIDR)*, 2005.

[86] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff i've seen: A system for personal information retrieval and re-use. In *Proc. of the Intl. SIGIR Conf. on Research and Development in Information Retrieval*, 2003.

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

[88] H. Ebel, L. I. Mielsch, and S. Bornholdt. Scale-free topology of email networks. *Physical Review E 66*, 2002.

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

[90] E. 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.

[91] N. Eiron and K. S. McCurley. Untangling compound documents on the web. In *Proc. of the 14th ACM Conference on Hypertext and Hypermedia*, 2003.

[92] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the web frontier. In *Proceedings of 13th International World Wide Web Conference*, 2004.

[93] G. Erkan and D. Radev. Lexpagerank: prestige in multidocument text summarization. In *Proc. of the 4th Intl. Conf. on Empirical Methods in Natural Language Processing*, 2004.

[94] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22, 2004.

[95] B. 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.

[96] G. Feng, T.-Y. Liu, Y. Wang, Y.Bao, Z. Ma, X.-D. Zhang, and W.-Y. Ma. Aggregaterank: Bringing order to web sites. In *In Proc. of the 29th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2006.

[97] P. Ferragina and A. Gulli. A personalized search engine based on web-snippet hierarchical clustering. In *Proc. of the 14th Intl. Conf. on World Wide Web*, 2005.

[98] S. Fertig, E. Freeman, and D. Gelernter. Lifestreams: An alternative to the desktop metaphor. In *Proc. of the ACM Conf. on Human Factors in Computing Systems*, 1996.

[99] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases*, 2004.

[100] M. Fisher and R. M. Everson. When are links useful? experiments in text classification. In *Proc. of the European Conf. on Information Retrieval*, 2003.

[101] D. Fogaras and B. Racz. Scaling link based similarity search. In *Proc. of the 14th Intl. World Wide Web Conf.*, 2005.

[102] 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.

[103] E. Garfield. Citation indexes for science. *Science*, 122:108–111, 1955.

[104] E. Garfield. Science citation index: A new dimension in indexing. *Science*, 144:649–654, 1964.

[105] E. Garfield. Citation analysis as a tool in journal evaluation. *Science*, 178:471–479, 1972.

[106] E. Garfield. *Citation Indexing: Its Theory and Application in Science, Technology and Humanities*. ISI Press, 1983.

[107] S. Gauch, J. Chaffee, and A. Pretschner. Ontology-based personalized search and browsing. *Web Intelli. and Agent Sys.*, 1(3-4):219–234, 2003.

[108] D. Geer. Will new standards help curb spam? *IEEE Computer*, pages 14–16, Feb. 2004.

[109] 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.

[110] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, 1998.

[111] D. Gleich, L. Zhukov, and P. Berkhin. Fast parallel PageRank: A linear system approach. Technical report, Yahoo! Research Labs, 2004.

[112] J. Golbeck and J. Hendler. Reputation Network Analysis for Email Filtering. In *Proc. of the Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, USA, July 2004.

[113] A. Gray and M. Haahr. Personalised, Collaborative Spam Filtering. In *Proc. of the Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, USA, July 2004.

[114] Z. Gyöngyi and H. Garcia-Molina. Link spam alliances. In *Proc. of the 31st Intl. VLDB Conf. on Very Large Data Bases*, 2005.

[115] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *Proceedings of the Adversarial Information Retrieval held the 14th Intl. World Wide Web Conference*, 2005.

[116] Z. Gyöngyi, H. Garcia-Molina, and J. Pendersen. Combating web spam with trustrank. In *Proceedings of the 30th International VLDB Conference*, 2004.

[117] T. Haveliwala. Efficient encodings for document ranking vectors. In *Proc. of the Intl. Conf. on Internet Computing*, 2003.

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

[119] T. Haveliwala and S. Kamvar. The second eigenvalue of the google matrix. Technical report, Stanford University, 2003.

[120] T. H. Haveliwala. Efficient computation of PageRank. Technical report, Stanford University, 1999.

[121] D. Hawking, E. Voorhees, N. Craswell, and P. Bailey. Overview of the trec8 web track. In *Eighth Text REtrieval Conference*, 1999.

[122] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *Proc. of the 11th Intl. SPIRE Conf. on String Processing and Information Retrieval*, 2004.

[123] T. Hoffmann. Probabilistic latent semantic indexing. In *Proc. of the 22nd Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1999.

[124] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *Proc. of the European Semantic Web Conference*, 2006.

[125] B. A. Huberman and L. A. Adamic. Information dynamics in the networked world. *Complex Networks, Lecture Notes in Physics*, 2003.

[126] J. Hull and P. Hart. Toward zero-effort personal doc. management. *IEEE Computer*, 34(3):30–35, 2001.

[127] 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.

[128] Isode. Benchmark and comparison of spamassassin and m-switch anti-spam. Technical report, Isode, Apr. 2004.

[129] T. J., A. C., A. M. S., and K. D. R. The perfect search engine is not enough: A study of orienteering behavior in directed search. In *In Proc. of CHI*, 2004.

[130] K. Järvelin and J. Keklinen. Ir evaluation methods for retrieving highly relevant documents. In *Proc. of the 23th Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2000.

[131] G. Jeh and J. Widom. Scaling personalized web search. In *Proc. of the 12th Intl. World Wide Web Conference*, 2003.

[132] K. S. Jones, S. Walker, and S. Robertson. Probabilistic model of information retrieval: Development and status. Technical report, Cambridge University, 1998.

[133] S. Kamvar, T. Haveliwala, C. Manning, and G. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University, 2003.

[134] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Extrapolation methods for accelerating PageRank computations. In *Proc. of the 12th Intl. Conf. on the World Wide Web*, 2003.

[135] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proc. of the 12th Intl. Conf. on World Wide Web*, 2003.

[136] D. R. Karger, K. Bakshi, D. Huynh, D. Quan, and V. 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.

[137] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43, 1953.

[138] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. *SIGIR Forum*, 37(2):18–28, 2003.

[139] M. Kendall. *Rank Correlation Methods*. Hafner Publishing, 1955.

[140] L. Kerschberg, W. Kim, and A. Scime. A personalizable agent for semantic taxonomy-based web search. In *First International Workshop on Radical Agent Concepts*, volume 2564 of *LNCS*. Springer, 2002.

[141] M.-C. Kim and K.-S. Choi. A comparison of collocation-based similarity measures in query expansion. *Information Processing and Management*, 35(1):19–30, 1999.

[142] S.-B. Kim, H.-C. Seo, and H.-C. Rim. Information retrieval using word senses: root sense tagging approach. In *Proc. of the 27th Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2004.

[143] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[144] C. Kohlschtter, P. A. Chirita, and W. Nejdl. 9. using link analysis to identify aspects in faceted web search. In *Proc. of the Workshop on Faceted Search held at the 29th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2006.

[145] C. Kohlschtter, P. A. Chirita, and W. Nejdl. Efficient parallel computation of pagerank. In *Proc. of the 28th European Conference on Information Retrieval*, 2006.

[146] J. Kong, P. Boykin, B. Rezaei, N. Sarshar, and V. Roychowdhury. Let your CyberAlter Ego Share Information and Manage Spam. Technical report, University of California, USA, 2005. Preprint.

[147] R. Kraft and J. Zien. Mining anchor text for query refinement. In *Proc. of the 13th Intl. Conf. on World Wide Web*, 2004.

[148] R. Krovetz and W. B. Croft. Lexical ambiguity and information retrieval. *ACM Trans. Inf. Syst.*, 10(2), 1992.

[149] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the web for emerging cyber-communities. In *Proc. of the 8th Intl. Conf. on World Wide Web*, 1999.

[150] S. R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. The Web as a graph. In *Proceedings of the Symposium on Principles of Database Systems*, 2000.

[151] W. Lam, S. Mukhopadhyay, J. Mostafa, and M. Palakal. Detection of shifts in user interests for personalized information filtering. In *Proc. of the 19th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1996.

[152] A. M. Lam-Adesina and G. 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.

[153] A. N. Langville and C. D. Meyer. Deeper inside PageRank. *Internet Mathematics (to appear)*, 2004.

[154] A. N. Langville and C. D. Meyer. A reordering for the PageRank problem. Technical report, NCSU, 2004.

[155] A. N. Langville and C. D. Meyer. A survey of eigenvector methods of web information retrieval. *The SIAM Review*, 47(1):135–161, 2005.

[156] M. Lansdale. The psychology of personal information management. *Applied Ergonomics*, 19(1):55–66, March 1988.

[157] H. C. Lee and A. Borodin. Perturbation of the hyperlinked environment. In *Proceedings of COCOON, LNCS 2696*, 2003.

[158] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):387–401, 2000.

[159] R. Lempel and S. Moran. Rank-stability and rank-similarity of link-based web ranking algorithms in authority-connected graphs. *Inf. Retr.*, 8(2):245–264, 2005.

[160] M. Levene, T. Fenner, G. Loizou, and R. Wheeldon. A stochastic model for the evolution of the web. *Computer Networks*, 39:277–287, 2002.

[161] L. Li, Y. Shang, and W. Zhang. Improvement of hits-based algorithms on web documents. In *Proc. of the 11th Intl. Conf. on World Wide Web*, 2002.

[162] Y. Li, Z. A. Bandar, and D. McLean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Trans. on Knowledge and Data Eng.*, 15(4):871–882, 2003.

[163] F. Liu, C. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. *IEEE Trans. on Knowledge and Data Eng.*, 16(1):28–40, 2004.

[164] A. G. Maguitman, F. Menczer, H. Roinestad, and A. Vespignani. Algorithmic detection of semantic similarity. In *Proc. of the 14th Intl. Conf. on World Wide Web*, 2005.

[165] 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.

[166] C. Martindale and A. K. Konopka. Oligonucleotide frequencies in dna follow a yule distribution. *Computer and Chemistry*, 20(1):35–38, 1996.

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

[168] M. E. J. Newman, S. Forrest, and J. Balthrop. Email networks and the spread of computer viruses. *Physical Review E 66*, 2002.

[169] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Stable algorithms for link analysis. In *Proc. 24th Annual Intl. ACM SIGIR Conference*, 2001.

[170] L. Nie, B. Davison, and X. Qi. Topical link analysis for web search. In *In Proc. of the 29th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2006.

[171] M. of the CLEVER Project. Hypersearching the web. Technical report, IBM, 2003.

[172] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[173] S. M. Pahlevi and H. Kitagawa. Taxonomy-based adaptive web search method. In *Intl. Symp. on Information Technology*, 2002.

[174] G. Pandurangan, P. Raghavan, and E. Upfal. Using pagerank to characterize web structure. In *Proc. of the 8th Intl. COCOON Conf. on Computing and Combinatorics*, 2002.

[175] M. Perone. An overview of spam blocking techniques. Technical report, Barracuda Networks, 2004.

[176] F. Qiu and J. Cho. Automatic indentification of user interest for personalized search. In *Proc. of the 15th Intl. World Wide Web Conf.*, 2006.

[177] Y. Qiu and H.-P. Frei. Concept based query expansion. In *Proc. of the 16th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1993.

[178] D. Quan and D. Karger. How to make a semantic web browser. In *Proc. of the 13th Intl. World Wide Web Conf.*, 2004.

[179] D. Rafiei and A. O. Mendelzon. What is this page known for? Computing web page reputations. In *Proceedings of the 9th International World Wide Web Conference*, 2000.

[180] P. Resnick and H. Varian. Recommender Systems. *Commun. ACM*, 40(3):56–58, 1997.

[181] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *Proceedings of the 2nd International Semantic Web Conference*, 2003.

[182] M. Richardson, A. Prakash, and E. Brill. Beyond pagerank: Machine learning for static ranking. In *Proc. of the 15th Intl. World Wide Web Conf.*, 2006.

[183] M. Ringel, E. Cutrell, S. Dumais, and E. Horvitz. Milestones in time: The value of landmarks in retrieving information from personal stores. In *INTERACT*, 2003.

[184] G. Roberts and J. Rosenthal. Downweighting tightly knit communities in world wide web rankings. *Advances and Applications in Statistics (ADAS)*, 3:199–216, 2003.

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

[186] I. Ruthven. Re-examining the potential effectiveness of interactive query expansion. In *Proc. of the 26th Intl. ACM SIGIR Conf. on Research and development in informaion retrieval*, 2003.

[187] H. Sakagami and T. Kamba. Learning personal preferences on online newspaper articles from user behaviors. In *Proceedings of the 6th International World Wide Web Conference*, 1997.

[188] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[189] K. Sankaralingam, S. Sethumadhavan, and J. C. Browne. Distributed pagerank for p2p systems. In *Proc. of the 12th IEEE Intl. Symp. on High Performance Distributed Computing (HPDC)*, 2003.

[190] T. Sarlos, A. A. Benczur, K. Csalogany, D. Fogaras, and B. Racz. To randomize or not to randomize: Space optimal summaries for hyperlink analysis. In *Proc. of the 15th Intl. World Wide Web Conf.*, 2006.

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

[192] L. Sauermann and S. Schwarz. Gnowsis adapter framework: Treating structured data sources as virtual rdf graphs. In *Proc. of the 4th Intl. Semantic Web Conf.*, 2005.

[193] T. H. H. Sepandar D. Kamvar and G. H. Golub. Adaptive methods for the computation of PageRank. Technical report, Stanford University, 2003.

[194] C. Shah and W. B. Croft. Evaluating high accuracy retrieval techniques. In *Proc. of the 27th Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2004.

[195] S. Shi, J. Yu, G. Yang, and D. Wang. Distributed page ranking in structured p2p networks. In *Proceedings of the 2003 International Conference on Parallel Processing*, 2003.

[196] H. A. Simon. On a class of stew distribution functions. *Biometrika*, 42:425–440, 1955.

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

[198] C. Soules and G. Ganger. Connections: using context to enhance file search. In *SOSP*, 2005.

[199] M. Speretta and S. Gauch. Personalizing search based on user search histories. In *Proc. of the 13th Intl. ACM CIKM Conf. on Information and Knowledge Management*, 2004.

[200] E. Spertus. Parasite: Mining structural information on the web. In *Proc. of the 6th Intl. World Wide Web Conf.*, 1997.

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

[202] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proc. of the 13th Intl. World Wide Web Conf.*, 2004.

[203] D. Sullivan. The older you are, the more you want personalized search, 2004. http://searchenginewatch.com/searchday/article.php/3385131.

[204] J. Teevan, S. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *Proc. of the 28th Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2005.

[205] J. A. Tomlin. A new paradigm for ranking pages in the world wide web. In *Proceedings of 12th International World Wide Web Conference*, 2003.

[206] P. Tsaparas. Using non-linear dynamical systems for web searching and ranking. In *Proceedings of the PODS Conference on Principles of Database Systems*, 2004.

[207] T. Upstill, N. Craswell, and D. Hawking. Predicting fame and fortune: Pagerank or indegree? In *Proceedings of the ADCS Australasian Document Computing Symposium*, 2003.

[208] E. Volokh. Personalization and privacy. *Commun. ACM*, 43(8), 2000.

[209] E. M. Voorhees. Query expansion using lexical-semantic relations. In *Proc. of the 17th Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 1994.

[210] E. M. Voorhees. The trec robust retrieval track. *SIGIR Forum*, 39(1), 2005.

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

[212] S.-C. Wang and Y. Tanaka. Topic-oriented query expansion for web search. In *Proc. of the 15th Intl. Conf. on World Wide Web*, 2006.

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

[214] J. B. Winer. *Statistical principles in experimental design.* McGraw Hill, 1962.

[215] G. Wittel and S. Wu. On Attacking Statistical Spam Filters. In *Proc. of the Conference on Email and Anti-Spam (CEAS)*, Mountain View, CA, USA, July 2004.

[216] B. Wu and B. Davison. Identifying link farm spam pages. In *Proceedings of the 14th World Wide Web Conference*, 2005.

[217] B. Wu and B. Davison. Undue influence: Eliminating the impact of link plagiarism on web search rankings. Technical report, LeHigh University, 2005.

[218] J. Wu and K. Aberer. Using SiteRank for decentralized computation of web document ranking, 2003.

[219] J. Xu and W. B. 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*, 1996.

[220] E. Yom-Tov, S. Fine, D. Carmel, and A. Darlow. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *Proc. of the 28th Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2005.

[221] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Proc. of the 12th Intl. Conf. on World Wide Web*, 2003.

[222] G. U. Yule. *Statistical Study of Literary Vocabulary*. Cambridge University Press, 1944.

[223] H. Zhang, A. Goel, R. Govindan, K. Mason, and B. van Roy. Improving eigenvector-based reputation systems against collusions. In *Proceedings of the 3rd Workshop on Web Graph Algorithms*, 2004.

[224] C. Ziegler and G. Lausen. Spreading activation models for trust propagation. In *Proceedings of the IEEE International Conference on e-Technology, e-Commerce, and e-Service*, 2004.

[225] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.

Walter-Gieseking-Str. 13
D-30159, Hannover
Germany

Tel: +49.176.264.854.24
E-Mail: chirita@l3s.de
Web: www.l3s.de/~chirita

Paul - Alexandru Chirita

Born on the 7th of July, 1980, in Bucharest, Romania.

KEY QUALIFICATIONS

- 4+ years experience with search engine technology applied both on the Web and on PC desktops.
- Good knowledge of data structures and algorithms, with focus on real world applications of Information Retrieval. Proficient in link analysis ranking and graph specific algorithms.
- Main current research interests cover personalized web search, link spam detection, and desktop search. Other interests include relevance feedback, word sense disambiguation and document clustering.
- Abstract thinker, with good problem solving and organizational skills.
- Excellent skills in project management, supervision, technical writing, as well as oral communication.
- Creative, success driven team player who continuously meets and exceeds goals.

EDUCATION

2003 – 2006 University of Hannover, Hannover, Germany

- Ph.D. in Information Retrieval (24.5.2007): Emerging Applications of Link Analysis for Ranking.
- Referees: Prof. Dr. Heribert Vollmer (head), Prof. Dr. Wolfgang Nejdl (thesis supervisor), Prof. Dr. Ricardo Baeza-Yates (external referee), Prof. Dr. Klaus Jobmann (secondary referee).

1998 – 2003 "Politehnica" University, Bucharest, Romania

- M. S. in Artificial Intelligence and Computer Graphics (Thesis: "Personalized Web Search").
- B. S. in Computer Science (Thesis: "Analysis of the S@TML Language to Provide Support for JavaCard Applets").
- Graduated within top 1% and received the maximum grade for both graduation theses.

2000 – 2001 Ecole Polytechnique, Paris, France

- EU funded scholarship. Awarded the "Très honorable" distinction (maximum possible) for my results during the sponsored year.

ADDITIONAL COURSES

- ACM SIGIR 2004-2006: Tutorials on Web Search Architectures (By K.-M. Risvik and M. Hearst), Database Information Retrieval (By R. Baeza-Yates and M. Consens), XML Retrieval (By R. Baeza-Yates and N. Fuhr), Language Modeling (By ChenXiang Zhai) and Multimedia Information Retrieval (By D. Ponceleon and M. Slaney).
- WEB-BAR 2004: Bertinoro Advanced Summer School in Web Information Retrieval and Mining, Bertinoro, Italy (By P. Raghavan and H. Schütze).
- Project Management Certified Professional (By Brainbench, 2006).

SELECTED PUBLICATIONS

- **Using ODP Metadata to Personalize Search**. Paul - Alexandru Chirita, Wolfgang Nejdl, Raluca Paiu, Christian Kohlschütter. In Proc. of the 28th ACM Intl. SIGIR Conference on Research and Development in Inf. Retr. (2005).
- **Site Level Noise Removal for Search Engines**. Andre Carvalho, Paul - Alexandru Chirita, Edleno Silva de Moura, Pavel Calado, Wolfgang Nejdl. In Proc. of the 15th Intl. World Wide Web Conference (2006).
- **Summarizing Local Context to Personalize Global Web Search.** Paul - Alexandru Chirita, Claudiu Firan, Wolfgang Nejdl. In Proc. of the 15th ACM Intl. CIKM Conf. on Information and Knowledge Management (2006).
- **MailRank: Using Ranking for Spam Detection**. Paul - Alexandru Chirita, Jörg Diederich, Wolfgang Nejdl. In Proc. of the 14th ACM Intl. CIKM Conference on Information and Knowledge Management (2005).
- **P-TAG: Large Scale Automatic Generation of Personalized Annotation Tags for the Web.** Paul - Alexandru Chirita, Stefania Costache, Siegfried Handschuh, W. Nejdl. In Proc. of the 16th Intl. World Wide Web Conf. (2007).

- **Analyzing User Behavior to Rank Desktop Items.** Paul - Alexandru Chirita, Wolfgang Nejdl. In Proc. of the 13th Intl. SPIRE Symposium on String Processing and Information Retrieval (2006).
- Additional 21 published papers and book chapters on Personalized Web Search, Desktop Search, Collaborative Filtering, as well as Peer-to-Peer Search and Publish/Subscribe Systems.

BOOKS

- **Data Structures and Algorithms in C++ using the STL**. Valeriu Iorga, Cristian Opincaru, Corina Stratan, Paul - Alexandru Chirita. PoliRom Publishing House, Bucharest, Romania, 2005; 352 pages, 16 x 23 cm.
- **C and C++ Programming Problems**. Valeriu Iorga, Paul - Alexandru Chirita, Corina Stratan, Cristian Opincaru. Niculescu Publishing House, Bucharest, Romania, 2004; 256 pages, 16 x 23 cm.
- **Computer Networks, Fourth Edition**. Andrew S. Tanenbaum. Co-translated the book into Romanian.

RELEVANT EXPERIENCE

2003 – 2007 **L3S Research Center, Hannover, Germany**

Researcher / Project Manager

- Previously employed as an intern (03.2003 – 08.2003), junior researcher (08.2003 – 03.2004), and team manager within the local iSearch project on designing personalized search algorithms (03.2004 – 10.2005)
- Manager of the L3S team within the EU Nepomuk project on building the Social Semantic Desktop, as well as co-author of its funding proposal (approximately 11.5 Mio. Euro).
- Served as a technical consultant for other projects in order to provide Information Retrieval specific expertise.
- Supplied the institute with new high quality researchers by acquiring additional partners and developing regular recruiting sessions with them.

2006 **Yahoo! Research Europe, Barcelona, Spain**

Visiting Researcher

- Worked with Prof. Dr. Ricardo Baeza-Yates on Author Ranking algorithms for the World Wide Web.

2005 **Federal University of Amazonas, Manaus, Brazil**

Visiting Researcher

- Worked with Prof. Dr. Edleno de Moura (creator of Google Brazil) on Link Noise Removal for Search Engines.

2004 – 2006 **University of Hannover, Hannover, Germany**

Teaching Assistant (Internet Technologies)

2002 – 2003 **National Center for Information Technology, Bucharest, Romania**

Research Assistant (EU Codestar Project, C++ OOP Tutor)

2001 – 2003 **"Politehnica" University, Bucharest, Romania**

Teaching Assistant (C/C++ Programming, Data Structures and Algorithms, Numeric Algorithms, Computer Systems)

2001 **Schlumberger Industries, Paris, France**

Wireless Engineering Intern

OTHER ACTIVITIES

- Fluent in English, French, German, and Romanian. Some knowledge of Spanish.
- Awarded several prizes, including the *Best Romanian Undergraduate Research Project of the Year* in 2000.
- Awarded a (merit) scholarship by the "Politehnica" University of Bucharest for the prominent professional results during all five years of study.
- Since 1994, I have always been a member (or leader) of different, official and unofficial student organizations.
- Hobbies include history, watching movies, as well as playing tennis and basketball.