

SIGIR 2010

Geneva, Switzerland
July 19-23, 2010

DESKTOP SEARCH

Workshop of the 33rd Annual International
ACM SIGIR Conference
*on Research and Development
in Information Retrieval*

Organised by
David Elsweiler
Gareth J.F. Jones
Liadh Kelly
Jaime Teevan



Association for
Computing Machinery

SIGIR
Geneva 2010

Preface

These proceedings contain details on the invited talks and the papers presented at the SIGIR 2010 Workshop on Desktop Search (Understanding, Supporting, and Evaluating Personal Data Search), Geneva, Switzerland, 23 July, 2010.

Despite recent research interest, desktop search is under-explored compared to other search domains such as the web, semi-structured data, or flat text. Even with the availability of several new desktop search tools, users are more successful finding information through browsing their personal collections and subsequently show preference for this approach. Problems with existing desktop search tools include performance issues, an over-reliance on good query formulation, and a failure to fit within the user's work flow or the user's mental model. As the available storage for desktop collections becomes cheaper and more plentiful and new media types continue to appear, the size and types of items stored in personal collections is growing rapidly. The need for effective methods to integrate the interaction experience between different information types is becoming ever more pressing. The aim of this workshop is to bring together people interested in desktop search with the goal of fostering collaborations and addressing the challenges faced in this area.

We would like to thank ACM and SIGIR for hosting the workshop. Thanks also go to the program committee, invited speakers and paper authors, without whom there would be no workshop.

July 2010

David Elsweiler
Gareth J.F. Jones
Liadh Kelly
Jaime Teevan

Organisation

Program Chairs

David Elswailer (University of Erlangen, Germany)
Gareth J. F. Jones (Dublin City University, Ireland)
Liadh Kelly (Dublin City University, Ireland)
Jaime Teevan (Microsoft Research Redmond, USA)

Program Committee

Leif Azzopardi (University of Glasgow, UK)
Ofer Bergman (Sheffield University, UK)
Daragh Byrne (Dublin City University, Ireland)
Rob Capra (University of North Carolina, USA)
Yi Chen (Dublin City University, Ireland)
Sergey Chernov (University of Hanover, Germany)
Bruce Croft (University of Massachusetts, USA)
Ed Cutrell (Microsoft Research, India)
Susan Dumais (Microsoft Research, USA)
Karl Gyllstrom (Katholieke Universiteit Leuven, Belgium)
Diane Kelly (University of North Carolina, USA)
Yukun Li (Renmin University, China)
Ian Ruthven (University of Strathclyde, UK)

Table of Contents

Preface.....	I
Organisation.....	II
Table of Contents.....	III

Invited Speakers

Stuff I've Seen: Retrospective and Prospective Views..... <i>Susan Dumais</i>	1
The Future of Desktop Search..... <i>Gregory Grefenstette</i>	2

Presentations

Detecting Contexts on the Desktop Using Bayesian Networks..... <i>Stefania Costache, Julien Gaugaz, Ekaterini Ioannou, Claudia Niederée, Wolfgang Nejdl</i>	3
Not Gone, but Forgotten: Helping Users Re-Find Web Pages by Identifying those which are most likely to be Lost..... <i>Karl Gyllstrom, Elin Rønby Pedersen</i>	7
deskWeb2.0: Combining Desktop and Social Search..... <i>Sergej Zerr, Elena Demidova, Sergey Chernov</i>	9

Posters

Memory Support for Desktop Search..... <i>Yi Chen, Liadh Kelly, Gareth J.F. Jones</i>	13
Using Personal Information Visualization for Document Retrieval..... <i>Paulo Gomes, Sandra Gama, Daniel Gonçalves</i>	17
Towards Task-Organised Desktop Collections..... <i>Yukun Li, David Elswailer, Xiaofeng Meng</i>	21
History Structure for Exploring Desktop Data..... <i>Harumi Murakami</i>	25
Improving Re-Finding upon Work Resumption..... <i>Thorsten Prante, Jens Sauer, Albrecht Schmidt</i>	27
Can Maps Provide the Answer to Desktop "Search"?..... <i>Roy A. Ruddle</i>	29
Metasearch Tools for Desktop Search..... <i>Paul Thomas, David Hawking</i>	33

Stuff I've Seen: Retrospective and Prospective Views

Susan Dumais
Microsoft Research
Redmond, WA

Biography

Susan Dumais is a Principal Researcher and manager of the Context, Learning and User Experience for Search (CLUES) Group at Microsoft Research. She has been at Microsoft Research since 1997 and has published widely in the areas of human-computer interaction and information retrieval. Her current research focuses on personal information management, user modeling and personalization, novel interfaces for interactive retrieval, and implicit measures of user interest and activity. Much of this work involves themes of interest to the Desktop Search Workshop -- understanding and supporting people in re-finding information they have previously created or seen (e.g., Stuff I've Seen, Personal Narratives), and using rich user models of interaction history to improve ranking and presentation (e.g., Milestones, Phlat, PSearch). Susan has worked closely with several Microsoft groups (Bing, Windows Desktop Search, SharePoint Portal Server, and Office Online Help) on search-related innovations. Prior to joining Microsoft Research, she was at Bellcore and Bell Labs for many years, where she worked on Latent Semantic Indexing (a statistical method for concept-based retrieval), combining search and navigation, individual differences, and organizational impacts of new technology.

Susan has published more than 200 articles in the fields of information science, human-computer interaction, and cognitive science, and holds several patents on novel retrieval algorithms and interfaces. She is Past-Chair of ACM's Special Interest Group in Information Retrieval (SIGIR), and serves on several editorial boards, technical program committees, and government panels. She was elected to the CHI Academy in 2005, an ACM Fellow in 2006, and received the SIGIR Gerard Salton Award for Lifetime Achievement in 2009. Susan is an adjunct professor in the Information School at the University of Washington, and has been a visiting faculty member at Stevens Institute of Technology, New York University, and the University of Chicago.

Additional information is available at:
<http://research.microsoft.com/~sdumais>

The Future of Desktop Search

Gregory Grefenstette
Exalead
Paris, France

Biography

Gregory Grefenstette is the Chief Science Officer of Exalead, the European leader in Search Based Applications. Grefenstette, who received his Ph.D. in Computer Science from the University of Pittsburgh, has been a fixture of the information retrieval and natural language processing fields since 1988.

A highly cited author and frequent member of international conference committees, he possesses 15 patents in the U.S. including the design of a photocopier for cross language information retrieval (US 6,396,951), for finding experts in a company by mining Web usage (US 6,446,035) and for creating documents that enrich themselves (US 6,732,090).

Most recently, Grefenstette worked as a senior expert at CEA (the Atomic Energy Commission - an applied research center in France). Before that, he worked as an engineer at Honeywell Bull, taught at the University of Tours and worked as a principal scientist at Xerox Research and Clairvoyance Corporation.

Detecting Contexts on the Desktop Using Bayesian Networks

Stefania Costache, Julien Gaugaz, Ekaterini Ioannou, Claudia Niederée, Wolfgang Nejd
L3S Research Center, Appelstr. 9A, 30169 Hannover, Germany
lastname@l3s.de

ABSTRACT

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

Categories and Subject Descriptors

H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing

General Terms

Algorithms

Keywords

User Context, Desktop, Bayesian Network, Evidences

1. INTRODUCTION

Recent Personal Information Management (PIM) projects focused on improving the management of desktop resources (e.g., files, emails), going beyond the functionality of commercial desktop search engines. Systems such as Haystack [13], Stuff I've Seen [3], and NEPOMUK [10] proved that automatically extracting and maintaining rich information describing desktop objects is feasible.

The next step in PIM is to provide better task support on the desktop, building upon the extracted information. Also, the borderline between desktop and Web deserves additional attention, since the personal information is no longer managed on the desktop only, but on the "Virtual Personal Desk-

top", an extension of the user's desktop on the Web. Our innovative approach for detecting (working) contexts serves this future PIM direction.

Task solving requires a set of related people, ideas, tools and resources, which are called working or task context. For the desktop, we define such a context as the collection of resources that the user uses to solve one task. By processing these resources we collect time-related similarity evidences. Also, identifying file similarities using text similarity techniques was found successful in many areas (e.g., [9] build on them for document classification). Our approach intelligently combines a variety of such evidences (textual and non-textual) to determine the working context. The resulting, improved knowledge about the user's context can be used, for example, to facilitate the user with the desktop resources she needs for the current task, as well as for targeted refinement of user profiles.

More specifically, we focus on identifying evidences supporting possible file-to-context assignments, based on the content of the files on the desktop, as well as the time connections between them. A Bayesian Network (BN) is constructed to model the evidences, the possible file-to-context assignments, and the interdependencies between them. We then use the BN to infer which files belong to which contexts.

Our main contributions are in identifying useful evidences for context detection (Section 3), evaluating their usefulness in preliminary experiments (Section 5), and creating a BN for successfully identifying desktop contexts (Section 4). Furthermore, we discuss related work in Section 2, and conclude with ideas and future work in Section 6.

2. RELATED WORK

For framing our approach, we present related work on the evidences used for analyzing desktop user behavior. These evidences mainly come from text analysis and implicit feedback about the user's desktop activities. Such evidences are used to predict patterns of user behavior, as well as for other applications on the desktop.

Desktop Usage Analysis. Desktop usage behavior has been thoroughly analyzed in many studies. For example, Malone [8] used interviews to analyze the way professional and clerical office workers organize information in their desks and offices - the *filers* and *plers* paradigm. This work is orthogonal to ours, as we also analyze desktop user activity, but we focus on file access distribution, rather than storage behavior. Also, [1] suggested that the way information is used on the desktop should also be the primary determinant of the way it will be organized, stored and retrieved. We

rely on the same idea: the user’s way of interacting with information on the desktop should deliver good evidences for identifying the user’s contexts.

User Activity & Implicit Feedback. Although implicit feedback is useful for predicting the importance of a resource for its user, there is almost no work dedicated to using it to group documents on the desktop, and use them as working contexts as we do in our approach. [11] discussed the different types of behavioral patterns that can occur on the desktop, mainly based on analyzing the time spent on one resource (reading time durations). A strong correlation is proven between the reading time and the importance of a document for the user. This supports our approach, since we log the focus of windows displaying resources, which we consider as reading time. They also state that behavior evidences are hardly taken into account in present research, and even less effort is put into combining this with content-based representation, the approach we envision. [4] reinforced these observations by discovering that they can be further improved by also logging rare but very meaningful desktop events, such printing a document indicating special document importance. [15] analyzed file activities to deduce links between files based on temporal locality in access. These links are used to provide ranking based on different algorithms, as also used in [5], while we use them for detecting contexts. A good summarization on implicit feedback measures, categorization and usage can be found in [7].

Text Based Context Detection. [2] used applications such as word processors to extract keywords representative for the current user task, and used them for pro-actively presenting the task-related documents to the user. Scatter/Gather [6] used the *Fractionation algorithm* to perform text clustering on search results and automatically organize them into a given number of topic-coherent groups. This is related to context detection, since, a cluster of documents represents a type of a context for the user to find her search results within. [14] uses clustering based on document term vectors to help reorganize the desktop and classify new files. In our approach, we also use text properties for grouping files but into contexts not into folder structures and we also use additional evidences from the access behavior of the user.

3. CONTEXT DETECTION EVIDENCES

As presented in Section 2, many approaches focus on identifying user behavior using evidences gathered from the desktop. Our evidences are mainly extracted from user actions and summarized within behavioral patterns, which can then be used to model user contexts. We present the different desktop evidences we collect and show their transformation into similarity measures which we use to construct the BN.

Textual Properties (T). Many approaches consider grouping together files (e.g., clustering) based solely on their textual properties. We follow [14], which focuses on desktop resources, and represent each document as a vector of TFx-IDF coordinates. Cosine similarity (or angle distance) between such vectors is used to model the similarity of desktop resources. When the vector similarity is small, we consider the documents similar, as done in Information Retrieval.

Usage Analysis (UA). Usage analysis refers to information when a file is active for a user, along with the file access times. The collected access times are used to compute the distance between any two resources using the sequence and session activity similarities described below.

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

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

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

$$g_r(s) = \sum_{s_i \in S} B_p(s - s_i) = \sum_{s_i \in S} \binom{N}{s - s_i} p^{s-s_i} (1-p)^{N-(s-s_i)} \quad (2)$$

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

$$\text{winCC}(f_a(t), f_b(t)) = \frac{\text{cov}(f_a(t), f_b(t))}{\text{std}(f_a(t)) \cdot \text{std}(f_b(t))}, \quad (3)$$

and the sequence similarity is:

$$\text{seqCC}(g_a(t), g_b(t)) = \frac{\text{cov}(g_a(t), g_b(t))}{\text{std}(g_a(t)) \cdot \text{std}(g_b(t))}, \quad (4)$$

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

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

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

Files Opened Concurrently (FOC). According to related studies (Section 2), when several resources are accessed in parallel at the same time they are (to some extent) related. UA logs the switching between resources, and FOC just considers the resources simultaneously accessed (resources displayed by a window at a given time), information not necessarily captured by UA.

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

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

Note that files sharing a common path prefix are also, in a restricted sense, in the same folder, i.e., they have a common

prefix. The fact that some files are in subdirectories of the common directory indicates however a lower probability to belong to the same context.

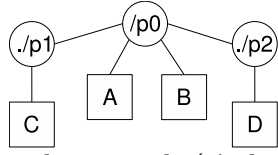


Figure 1: Hierarchy example (circles are directories and squares files).

Intuitively, files *A* and *B* in Figure 1 have the highest probability to belong to the same context. Files *A* and *C* are a bit less probable to belong to the same context, since they don't have exactly the same path: they are both under */p0* but *C* is also under */p1*. And files *C* and *D* are less probable to belong to the same context since they are in two different subcontexts (i.e., subdirectories) of */p0*.

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

4. THE CONTEXT BAYESIAN NETWORK

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

The BN is constructed from the following nodes:

- **Evidence Nodes** describe the type of evidences we have for each file. They can be text, UA, FOC, or FOH, as described in Section 3. These nodes represent the relation of two files with a specific evidence type.

- **Directly Related Nodes** describe the relationship between two files, based on Evidence Nodes. It is an unification of evidences that can be gathered for two files. Therefore, direct links between a Directly Related Node and their Evidence Nodes exist (see node "F1_related_F5" in Figure 2).

- **Inferred Related Nodes** also represent a relation between two files, but an indirect one. This type of nodes is suited for pairs of files for which we don't have direct evidence (text, UA, FOC, or FH), but their relationship can be transitively inferred from the Directly Related Nodes. For example, if we have two Directly Related Nodes which express the relationship between (F1, F2) and (F1, F5) (see Figure 2), we infer relationship (F2, F5) in this type of node.

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

- **Inferred Context Nodes** also rely on a transitive relationship inferred when we know there is a relationship (direct or inferred) between two files – (F1, F5), and we also know that one file belongs to a context – (F5, C1). Then we can easily infer that F1 should also belong to the same context – (F1, C1) (see Figure 2).

Construction of the BN. In order to construct the BN, we first collect the evidences that we have for the similarity

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

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

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

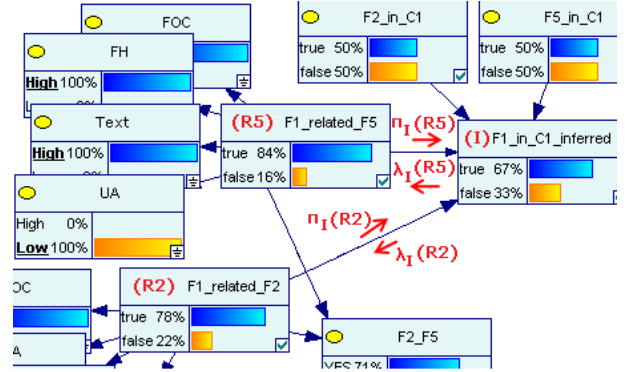


Figure 2: Small part of our BN.

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

Inference. When the BN is completed, we determine the probability of cause nodes (file to context assignment) given the probability of the observed effect (evidences), and identify which files belong to each context. This task is performed using Pearl's probabilistic inference (PI) [12]. At each step, an activated node *N* recomputes its own belief using messages collected from its parents ($\pi_N(P_i)$) and its children ($\lambda_N(C_j)$) (see Figure 2). Once the node has its belief, it sends messages to its parents ($\lambda_N(P_i)$) and its children ($\pi_C_j(N)$), which are then used when these nodes are activated. Once PI finishes, the Inferred Context Nodes pro-

vide probabilistic information about each file belonging to a context. Also, the information generated by the Inferred Related Nodes describes the files about which nothing could have been inferred relative to them belonging to a context. This allows us to identify new contexts which the user did not specifically name in her initial input.

5. EXPERIMENTS

We define a user desktop as all data stored by a user on a personal machine. This includes personal files (e.g., HTML, DOC, PDF, JPG), Web cache history, messenger history, emails. Upon their full text, standard text preprocessing techniques were performed: tokenization, stop words removal, stemming. We then computed the similarities for each pair of desktop resources, as described in Section 3.

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

Experimental Results & Comments

True positive evidences. We check if for two files that the user assessed as being in the same context (i.e., related), the evidences that we generate also support this fact – their values are higher than chosen thresholds, and therefore considered positive evidences. For each two such files, we consider the fraction of positive evidences from the total of evidences for a pair of files as the realScore. Then, we count the percent of the pairs of files from all files for which the realScore is higher or equal to 1, 0.75, 0.66, 0.5, 0.33, as shown in Table 1. These first results show that the generated evidences are correct, since at least 33% of the generated evidences for related files are shown to be positive, and 89.89% of the pairs of related files have all evidences positive.

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

Table 1: True positive evidences.

True negative evidences. This measure shows that the negative evidences (lower than a threshold) about two files are in correlation with the fact that the user said that two files are not related – one file is in one context and the other is not in the same context. We compute for all such pairs, the number of negative evidences, and divide it to the total number of evidences for the same pair of files and get again a realScore. Then, we compute the percentage of the pairs for which the realScore is higher or equal to 1, 0.75, 0.66, 0.5, 0.33, as shown in Table 2. This again supports the evidences that we generated, saying that at least 33% of the generated evidences for unrelated files are negative, and 76.87% of the pairs of unrelated files have all evidences negative.

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

Table 2: True negative evidences.

BN Performance. The user assessment was split into three parts, and two thirds were used for training the BN and one third used for evaluation. Then, the evaluation set was varied from within the three chunks of the user’s assessment.

For each of the three iterations, precision and recall were computed for each of the user clusters, and then averaged over all clusters. The preliminary results are supporting our ideas – precision was of 77.78% and recall 73.97%. However, further experiments are required to validate these results in a higher variety of settings: an increased number of users, various threshold values for choosing the positive evidences, as well as different initial probability table assignments in the BN, or an increased number of assessed files.

6. CONCLUSIONS AND FUTURE WORK

In this paper we presented innovative similarity measures used for detecting user contexts through a BN. Our preliminary experiments show promising results regarding the choice of evidences with respect to user assessments (both positive and negative), and also good results regarding the output of the BN and its capability of inferring user contexts. We are currently working on improving the results of our approach by incorporating additional evidences in the BN.

References

- [1] D. Barreau and B. Nardi. Finding and reminding: File organization from the desktop. In *ACM SIGCHI Bulletin*, 1995.
- [2] J. Budzik, K. J. Hammond, and L. Birnbaum. Information access in context. In *Knowl.-Based Syst.*, 2001.
- [3] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. Robbins. Stuff i’ve seen: a system for personal information retrieval and re-use. In *SIGIR*, 2003.
- [4] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve the search experience. In *Workshop on Implicit Measures of User Interests and Preferences, SIGIR*, 2003.
- [5] J. Gaugaz, S. Costache, P. A. Chirita, C. Firan, and W. Nejdl. Activity based links as a ranking factor in semantic desktop search. In *LA-WEB*, 2008.
- [6] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *SIGIR*, 1996.
- [7] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. 2003.
- [8] T. Malone. How do people organize their desks? implications for the design of office information systems. *TOIS*, 1983.
- [9] F. Mourão, L. C. da Rocha, R. B. Araújo, T. Couto, M. A. Gonçalves, and W. Meira. Understanding temporal aspects in document classification. In *WSDM*, 2008.
- [10] Nepomuk project. <http://nepomuk.semanticdesktop.org/>.
- [11] D. W. Oard and J. Kim. Modeling information content using observable behavior. In *ASIST*, 2001.
- [12] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.
- [13] D. Quan and D. Karger. How to make a semantic web browser. In *WWW*, 2004.
- [14] D. Rose, R. Mander, T. Oren, D. Ponceleon, G. Salomon, and Y. Wong. Content awareness in a file system interface: Implementing the ‘pile’ metaphor for organizing information. In *SIGIR*, 1993.
- [15] C. Soules and G. Ganger. Connections: using context to enhance file search. In *SOSP*, 2005.

Not gone, but forgotten: Helping users re-find web pages by identifying those which are most likely to be lost

Karl Gyllstrom
Department of Computer Science
Katholieke Universiteit Leuven
Leuven, Belgium
karl.gyllstrom@cs.kuleuven.be

Elin Rønby Pedersen
Google, Inc.
Mountain View, CA
USA
elinp@google.com

ABSTRACT

We describe LostRank, a project in its formative stage which aims to produce a way to rank results in re-finding search engines according to the likelihood of their being lost to the user. To this end, we have explored a number of ideas, including applying users' temporal document access patterns to determine the documents that are both important and have not been recently accessed (indicating greater potential for loss), understanding users' topical access patterns to determine the topics that are more unfamiliar and hence more difficult to re-find documents within, and assessing users' difficulties in originally finding documents in order to predict future difficulties in re-finding them. As a position paper, we use this as an opportunity to describe early work, invite collaboration with others, and further the case for the use of temporal access patterns as a source for assisting users' re-finding of personal documents.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]

General Terms: Human Factors

Keywords: Re-finding, ranking, log analysis

1. INTRODUCTION

Personal document collections grow constantly. Each day we access a significant number of new web pages, many of which we will probably never access again. One challenge is that finding a document within a large collection requires a specific query to distinguish the file from others in the collection. As time passes, our recollection of document specifics – with which we would formulate queries – decays. In other words, as time goes on, not only does our document collection grow larger – and hence harder to search – but our ability to issue good queries declines.

One area that deserves attention is the ranking function for search results, as a strong one can allow desktop search to produce good results for vague queries on large personal datasets. Additionally, it allows for a more aggressive expansion of users' queries to include topical or syntactic synonyms, as users are more likely to forget key terms (or use wrong terms) when re-finding documents accessed further into the past. Ranking is an important subject in re-finding because it addresses a fundamentally different problem than search for new information, and limits what can be imported

from mature domains such as web search. For example, on the web, algorithms like PageRank or HITS are effective ranking functions because they promote credible or authoritative pages, and when users seek answers to new questions, they desire answers that are most likely to be accurate. Credibility is defined by lots of incoming links from other credible pages, which has the effect of more highly ranking pages that are considered more important by the large community of web authors.

We argue that this is the opposite of what is desired for re-finding tasks. Though hyperlink structure is not present on users' filesystems, consider an analog assessment of importance, such as number of shortcuts, or proximity to the home or desktop directories. These qualities indicate that a document is quite important, and yet, they provide evidence that the document is unlikely to be lost, as it is readily accessible. Lost information tends to be that which is hidden within a difficult navigation path, such as within deeply nested directories or large files.

In our work, we have pursued ways to rank documents according to their likelihood of being "lost". In this context, we define lost documents to be those which a user has previously accessed, desires to access again, and is unable to find using traditional search methods, such as text-based desktop search. Classifying a document as lost is obviously a difficult and large endeavor. We have developed a few ideas which we are beginning to explore and evaluate, including page access patterns, topic access patterns, and difficulties surrounding the original document discovery. We describe these below, but first, let us summarize our use of personal web log data.

2. WEB LOG DATA AND ABSTRACTIONS

Our goal is to break a user's document activity into higher-level abstractions that allow us to better reason about it. In this work, we focus on web history because it is easy to extract (e.g., via Firefox), and, since it contains queries, it allows us to better understand information seeking behavior. We believe this approach could be extended to general document activity recording systems.

A web history is a time-ordered sequence of events, where an event is either a query, including query text, or a page click, including the URL and page contents. We process it using a two-fold approach: First, the history is separated into segments, where segments encompass a sequence of queries and page visits that occur within 5 minutes of each other. A segment roughly (though imperfectly) approximates a single task (e.g., searching for housing). Second,

the LDA topic detection algorithm [1] is run on the contents of the pages within these segments. These two approaches assign to each page a set of tasks and topics – including the relative strength of relationship between the page and each of its topics [2].

For each segment, we assign a difficulty assessment, which is measurement of the apparent difficulty of the information seeking task. We have selected a number of qualities, including number of queries, number of query reformulations (modifications of unsuccessful search attempts), length of session, number of queries for which no results are clicked (indicating poor queries), and average page view time. Pages within a segment inherit its difficulty score.

3. RANKING COMPONENTS

In this section we describe a few ranking components we have explored. After independently evaluating them we hope to combine them into a comprehensive ranking function. We envision adding more as this project matures.

3.1 Access patterns

As memory decays with time, the likelihood of a document being lost increases with the time since its last access. However, time-of-last-access alone is not sufficient to suitably rank documents. We use look beyond time-of-last-access to consider larger access patterns. For example, consider two pages that were last accessed by a user one month ago. Constrained to time-of-last-access, we would rank these pages as equally lost. Let us assume that one of the pages was first viewed at this time, while the other page has been accessed once per month for the last 2 years. We might reason that the latter page is less likely to be lost because its time-of-last-access is consistent with a larger pattern of access, and assign it a weaker rank.

Another case is we consider is when documents’ access patterns change. For example, a page that was very frequently accessed for a period of several months, but then not accessed at all for a year, has a pattern that we refer to as *dormant*. This pattern fits our definition of lost in that it indicates that the page was once important to the user (indicating that they may want to eventually use it again), and that the user’s familiarity with the page has declined (as evidenced by not being accessed for a long time).

3.2 Topic patterns

We extend the above notion to include topic, with the observation that users’ revisit patterns vary according to topic. For example, queries for code documentation might frequently be navigation-style queries for which the user has little difficulty finding relevant answers (e.g., looking up the Java `Set` class). Other topics, such as health, may involve more complex search processes where the answer to a question is more vague.

Our current implementation is to determine, for each page, the most closely linked LDA topics, and record an event for the topic at that point. This allows us to build an access pattern for each topic, and associate topical activity to each page. Pages with more dormant topics may be those which are more likely to be lost. The advantage of using topic is that we can reason about pages that the user has not accessed enough for a reliable pattern to emerge (e.g., the user only looks up code for Java class `String` once, but looks up code for Java classes routinely; it would therefore be as-

signed a weaker rank as the topic pattern indicates it is likely easily re-found).

3.3 Difficulty before original access

We consider the path a user takes to originally access a page, using the difficulty assessment described in Section 2. Repeated navigational queries – web queries that are intended to find a specific page (e.g., “ebay”) – suggest an easily re-found page. Pages discovered after long trails of queries and query reformulations indicate that the overarching task may have been more vague, or that the user lacked prior knowledge before the research task. We hypothesize that, as the latter are cases where the user’s understanding of the topic is weaker, the user’s recollection of terms from pages from difficult tasks will be worse, especially for the pages accessed later in the task. For example, a research path that began on energy-efficient buildings may have resulted in research on passive windows, the latter being a term less easily remembered if the user continues or restarts the research weeks or months later. Their query formulation may tend toward their original terminology rather than the terminology used in pages accessed after the task evolved.

4. CONCLUDING REMARKS

Most of the ideas described in this work originated from observations on a small number of very large query logs that volunteers offered for our use. We would like to evaluate them directly on a larger pool of user data, and invite the comments and participation of the community. In particular, we would like to see more research emphasis on personalized ranking in the context of re-finding.

There are a number of related works that have inspired this work. Several systems aim to improve document re-finding by tracing users’ desktop activity, for example, by detecting task relationships [3, 4]; our work would benefit from these systems’ tracing approaches, and allow us to integrate better task representations. The Re:search engine enhances web search by integrating previously accessed pages into search results for queries with similarity to previously issued queries [5]; we share a common goal, although we focus on determining which previously accessed pages to show to users.

References

- [1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.
- [2] E. R. Pedersen, K. Gyllstrom, S. Gu, and P. J. Hong. Automatic generation of research trails in web history. In *IUI '10*, pages 369–372, New York, NY, USA, 2010. ACM.
- [3] T. Rattenbury and J. Canny. CAAD: an automatic task support system. In *CHI '07*, pages 687–696, New York, NY, USA, 2007. ACM.
- [4] C. A. N. Soules and G. R. Ganger. Connections: using context to enhance file search. *SIGOPS Oper. Syst. Rev.*, 39(5):119–132, 2005.
- [5] J. Teevan. The re:search engine: simultaneous support for finding and re-finding. In *UIST '07*, pages 23–32, New York, NY, USA, 2007. ACM.

deskWeb2.0: Combining Desktop and Social Search

Sergej Zerr
L3S Research Center, Leibniz
University of Hanover
Appelstr. 9a, Hanover 30167,
Germany
zerr@L3S.de

Elena Demidova
L3S Research Center, Leibniz
University of Hanover
Appelstr. 9a, Hanover 30167,
Germany
demidova@L3S.de

Sergey Chernov
L3S Research Center, Leibniz
University of Hanover
Appelstr. 9a, Hanover 30167,
Germany
chernov@L3S.de

ABSTRACT

With availability of Web 2.0 platforms such as Flickr and YouTube, personal information is no longer locked within a user's desktop, but becomes increasingly distributed and shared across various online applications. In these settings it is important to provide a quick glance at the available personal resources and facilitate their search and selective sharing. This paper describes the challenges and requirements to be addressed in this context and presents deskWeb2.0 – an integrated environment which we currently implement towards this goal. We report the results of a small user study regarding effectiveness of such integration for different types of desktop search.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – Systems and Software

Keywords

Desktop Search, Social Search, Web 2.0.

1. INTRODUCTION

With availability of Web 2.0 platforms such as Flickr [7], YouTube [18] and Del.icio.us [5], personal space of information is no longer locked within a single desktop, but becomes increasingly distributed and selectively shared across various online applications. The Web 2.0 applications and social platforms are famous as convenient tools for sharing personal information. For example, recent report [10] shows that around 115 million bookmarks were available on the del.icio.us social bookmarking site alone in 2008. In these settings it is important to provide a quick glance not only at the user's files available locally, but also include resources shared online by the user and her friends in search results. Whereas each single Web 2.0 application is specialized in a set of predefined tasks, users would expect a single search interface over the entire set of distributed personal knowledge resources.

We illustrate this problem with a following example: Alice organizes her trip to the SIGIR 2010 conference in Geneva. She needs to retrieve relevant resources stored on her desktop, such as a sample form to authorize the trip and trip-related e-mail communication. Also, she would like to see hotel

recommendations from her colleagues and multimedia resources related to places of interest in Geneva visited by her friends. Finally, she is interested in links and news shared by the other conference participants. This task would require performing search on her desktop and within each relevant social application such as Youtube, Flickr, and Del.icio.us separately. Alternatively, Alice can use *deskWeb2.0* to retrieve all these resources at once. Figure 1 presents an overview of *deskWeb2.0*.



Figure 1. *deskWeb2.0* Overview

Majority of the existing desktop search applications do not support integrated search over shared social resources. A few applications such as Google Desktop Search [8] try to combine web and desktop search results, while do not support sufficient integration of search results obtained from user's accounts on social platforms. This lack of integration requires users to perform search in each social platform separately, which is a tedious task. In this paper we present *deskWeb2.0*, an integrated environment which gives its users a quick glance at the available personal resources independently of the hosting application. The contributions of this paper include: (i) a search algorithm over user's personal social network; (ii) a small-scale user study to assess how different desktop search tasks benefit from integration of social search results.

2. RELATED WORK

So far social search has not been addressed in conjunction with the desktop search problem. While modern desktop search applications allow to mix search results from the web and desktop (Google Desktop [8]) or index information on network drives (Windows Search [16], Autonomy IDOL Enterprise Desktop Search [1]), they do not search over user's Web 2.0 data, as until

now a major part of users' data was stored locally. Therefore, previous research in the area of "semantic desktop" is focused on extracting locally available metadata and storing it into a single RDF-based repository like in Haystack [13] and Gnowsist [14] systems. A recent approach implemented in the Beagle++ system [12] uses both desktop-located resources and external data proactively fetched from the Web. In contrast, our approach directly queries resources shared by the user and her friends on social services.

Social search systems allow searching for resources of different types, such as URLs, people, tags and their connections and offer ranking algorithms which take into account the structure of a social network. Hotho et al. in [9] developed ranking algorithms such as adapted PageRank and FolkRank which take network structure into account. Later, Bao et al. [2] presented alternative algorithms called SocialSimRank and SocialPageRank. Personalized ranking using social factors was considered by [3, 17]. In the current work we incorporate important ranking factors of social search to enable top-k processing.

One important task of *deskWeb2.0* is to provide an overview over the available results. To increase novelty of results and reduce the risk of user dissatisfaction, a number of schemes for diversifying results of document retrieval and database search have been proposed (e.g. [6, 15]). To give the user a quick glance of the available resources, in the current implementation we simply restrict the total number of results obtained from each service as well as the number of results retrieved from a particular user. In future work we plan to investigate alternative methods for results diversification.

3. CHALLENGES

When a large part of personal resources is distributed among heterogeneous social services, it becomes extremely difficult to provide satisfactory desktop search results. In this work we use the notion of *social search* to describe the search process over data gathered from user's personal networks in Web 2.0 applications, such as social bookmarking systems, blogs, forums, social network sites (SNSs), and others [3]. To integrate desktop and social search within *deskWeb2.0* we need to address several challenges discussed below.

Challenge 1: Technical and Semantic Interoperability

Both technical and semantic interoperability is required for authentication, authorization, sharing and search services of the connected platforms. Currently, such functionality is partially supported by the Web 2.0 tools using platform-dependent APIs. Given a large number of available services and possible software updates such integration becomes an essential technical problem and a laborious engineering task. Moreover, as different systems focus on specific shared data types and support different syntax, it becomes important to address these differences at a query transformation step. Finally, resources within social networks frequently change and require efficient update propagation to guarantee up-to-date search results. At the moment we address the

integration problem on a purely technical level and prepare a query for each service by applying service-specific heuristics.

Challenge 2: Ranking and Aggregation of Search Results

Resources from different social platforms differ in their relevance, quality, and relation to the user significantly. Furthermore, users often share sequences of similar resources, such as photo series, such that search results can contain (near-) duplicates or similar resources in different formats. Following the ideas developed in social search [4], the relevance of resources should be influenced by the distance within the personal network, i.e. resources of closely connected peers should be ranked higher compared to resources gathered from friend-of-a-friend (FOAF). Moreover, even if all resources in a sequence have similar relevance to the query, aggregated search results should rather provide an overview over the available options. Our initial implementation of *deskWeb2.0* takes into account the distance within the personal network to facilitate top-k processing and presents a fixed number of results from each relevant platform. To this end, search result diversification, which recently attracted a lot of attention in the context of Web- and database search [6, 15] can be considered in the future work.

Challenge 3: Privacy – Preserving Resource Sharing

Web 2.0 sharing platforms represent dynamic data sources and provide up-to-date visual information about locations, people, cultural events and travelling routes. With an increasing availability of publishing applications like iPhoto [11] for the mobile devices, these resources can be almost immediately uploaded and made available within the personal network of a user on the social platforms. As such resources can be of highly private nature, they need to be handled with care. A search system is required to assist users to automatically determine the level of privacy for a particular resource. As *deskWeb2.0* searches over resources already accessible to the user and does not store them in external indices, it does not violate user's privacy.

4. SEARCH ALGORITHM

To answer a query, *deskWeb2.0* gathers user's personal resources as well as resources from the user's social network available through FOAF relationships. We model an integrated user's network as a tree, where each edge represents a friendship relationship and each node represents a user. The root node is the querying user. The children of the root node are the direct friends of the user from each connected platform. To transform a social network graph into a tree we apply a greedy algorithm which selects the shortest paths within the graph. To retrieve up-to-date diverse search results, we implement a query propagation algorithm presented in Algorithm 1. Algorithm 1 traverses the tree in a breadth first manner. As a node can possibly contain either too many or too few results, the goal of Algorithm 1 is to obtain a balanced result set giving the user an overview over the available results. To decide on the number of results to be retrieved from a node n , it weights k in top-k with the relevance of the node n . In case the node does not contain enough results for a query, it propagates the remaining number of results to the n 's children.


```

getTopk(keywords, root, k, min_relevance, results){
  priorityQueue<relevance> queue;
  //compute the max number of results to be returned from the node
  root.maxresults=root.relevance * k;
  queue.enqueue (root);
  while (true) {
    node = queue.dequeue();
    //check relevance threshold
    if (node.relevance < min_relevance) break;
    node.results=node.query(keywords, node.maxresults);
    //collect results from the node
    results.add(node.results);
    if (results.size()>=k) break;
    for (friend: node.friends){
      //propagate remaining results to child's nodes
      friend.maxresults=node.maxresults-node.countresults;
      queue.enqueue(friend); } }

```

Algorithm 1. *deskWeb2.0* Search

The relevance of a resource in the integrated social network of *deskWeb2.0* depends not only on the content of the resource, but also on the global importance of its owner within the network and the strength of the relationship between the owner and the querying user. This relevance can be computed using Equation 1:

$$rel(r, n, q) = rel(n, n.network) \cdot rel(root \rightarrow n) \cdot rel(r.content, q), \quad (1)$$

where relevance of the resource r from node n with respect to query q is the relevance of the node n in the context of its social network $n.network$, $rel(root \rightarrow n)$ is the relevance of the node n with respect to the querying user $root$ and $rel(r.content, q)$ is the relevance of the content of r with respect to the query q . According to Equation 1, each node within user's network can be weighted independently of the query which supports efficient top-k query processing.

5. EVALUATION

To find out how users search personal resources using currently available tools we carried out a small questionnaire. We asked 22 graduate students from Computer Sciences department to tell us which tool they use to find resources on their desktop. Majority of the Windows users (10 out of 12) use native Windows Desktop Search tool; 3 participants uses other tools, such as Spotlight or "find" command for Linux; 9 users do not use desktop search tools. As our current implementation relies on Google Desktop Search to retrieve local search results, we had to limit our user study to few people who installed it.

In our evaluation of *deskWeb2.0* we focused on two research questions: **Question 1**: Does search over social services contributes to desktop search with respect to the relevance of results? **Question 2**: Which types of desktop search such as location, people and general information finding benefit from social search and which do not?

To answer these questions we performed a small user study. Our participants were five students from Computer Sciences department, who had Google Desktop Search tool installed. We

selected four tasks for the users to perform, each including three search types. Each task required the user to retrieve certain information from the integrated environment of *deskWeb2.0*:

T1. Collect information for a business trip; **T2**. Prepare a tutorial on a topic of interest; **T3**. Organize a short-distance weekend trip with friends/family; and **T4**. Organize a party. Each task included the following search types: **A**. Find contact details of a person; **B**. Find location information; and **C**. Find general information. For each task and search type, we asked users to issue a keyword query of their choice. For every query, we presented the user with two lists of results, one containing top-5 results from Google Desktop Search, and another list containing up to five results from each Web 2.0 service on which the user had an account. We asked users to rate the results on a 3-point scale as "relevant", "less relevant", or "non-relevant". The users had one or two active accounts on social services supported by our prototype so quantities of desktop and social results were comparable.

To answer **Question 1**, we computed the macro-averaged Normalized Discounted Cumulative Gain (nDCG) in three result lists: desktop, Web 2.0, and a merged list. To compute nDCG we ranked each list by TF-IDF scores. On Figure 2 we present the nDCG values for top-5 results averaged over the participants. As we can see from Figure 2, although absolute nDCG values of Web 2.0 results is lower than the values obtained by the desktop search, combination of desktop and social search results increases the gain of desktop search for all $k > 2$ by about 6% on average. We also report the results per each task T1 – T4 to see if there are any specific situations in which social search is useful. For readability reasons we split these results into two plots i.e. Figure 3 and Figure 4 and present only desktop and merged results. From the task-wise presentation we observe that tasks T1 and T3, both related to travelling preparations, only modestly benefit from merging with social search results. In contrast, task T2 about tutorial preparation shows stable and significant improvement over pure desktop search. Finally, party-planning task T4 shows about double nDCG improvement over regular desktop search. Therefore, we conclude that **Question 1** could be answered positively and search over social services significantly complements relevance of the desktop search.

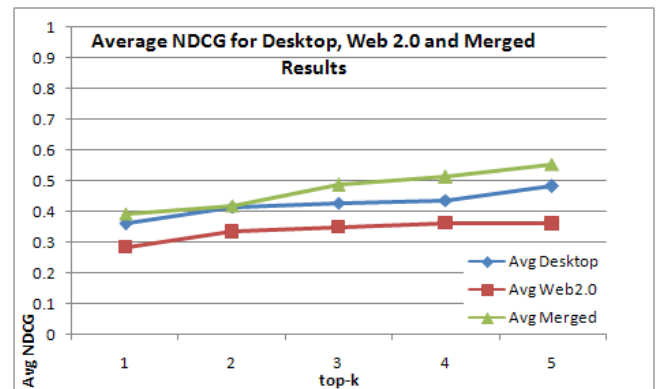


Figure 2. Average nDCG of all Tasks

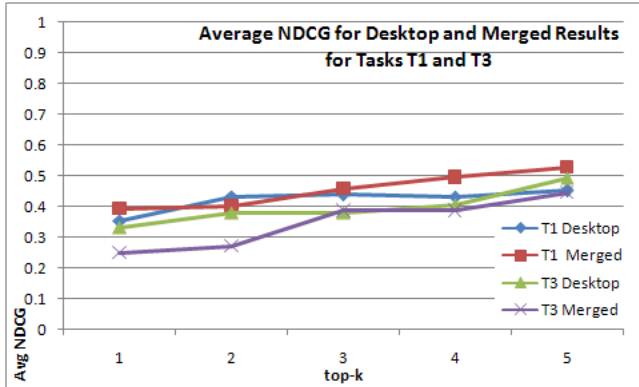


Figure 3. Average nDCG for Tasks T1 and T3

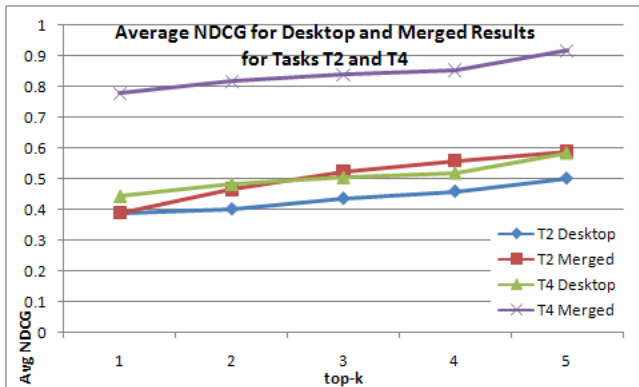


Figure 4. Average nDCG for Tasks T2 and T4

To answer **Question 2**, we considered nDCG in desktop and merged results for each type of search such as people (A), location (B) and general information (C) separately. Figure 5 presents nDCG results averaged over the users and search types. We observe that both people and general information search profit from the mixture of desktop and social search. nDCG for people search improved by about 17% and general information finding by 10%. On the contrary, nDCG of location search in the merged list decreased. Since nDCG of the social search results for location search was much lower than that of the desktop search their mixture did not provide any extra advantage. This result also explains the modest improvements in travelling-related tasks T1 and T3 where location search is important.

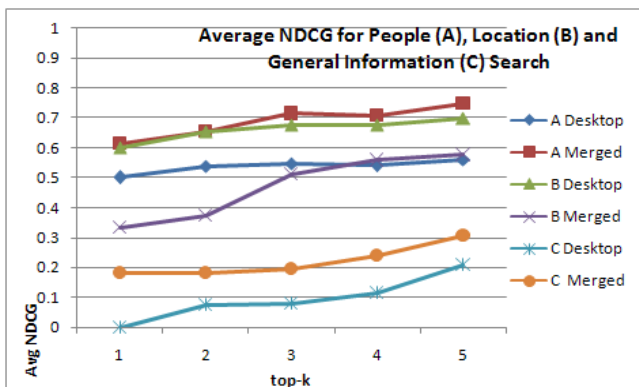


Figure 5. Average nDCG for three Types of Desktop Search

Our answer for the **Question 2** is that general information search benefits the most from social search, possibly, due to the low desktop search effectiveness. Desktop search for people finding is already very effective, but it is also significantly improved by the Web 2.0 results. Location search degrades when using social search results, we assume that users did not have relevant information for this search type in the current networks.

6. CONCLUSION

Nowadays, personal resources are no longer stored within a single desktop, but are increasingly shared across social platforms. This work presents the first steps towards integrating desktop search with social search over shared personal resources. We identified some challenges to be addressed and developed a sample *deskWeb2.0* application. A small user study demonstrated that search over social resources increases overall search accuracy. It also suggests that people finding and search for general information benefit from such integration, while search for locations is more effective using desktop search alone. In the future we plan to perform a larger user study. Also, we would like to investigate diversification of search results in this context.

7. REFERENCES

- [1] Autonomy IDOL Enterprise Desktop Search <http://www.autonomy.com/content/Products/enterprise-search/index.en.html>
- [2] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In WWW 2007.
- [3] M. Bender, T. Crecelius, M. Kacimi, S. Michel, T. Neumann, J. X. Parreira, R. Schenkel, and G. Weikum. Exploiting social relations for query expansion and result ranking. ICDE Workshops, 2008.
- [4] D. Carmel, N. Zwerdling, I. Guy, S. Ofek-Koifman, N. Har'El, I. Ronen, E. Uziel, S. Yogev, S. Chernov. Personalized social search based on the user's social network. In Proc. of CIKM 2009.
- [5] Delicious: <http://delicious.com/>
- [6] E. Demidova, P. Fankhauser, X. Zhou, W. Nejdl. DivQ: Diversification for Keyword Search over Structured Databases. In Proceedings of SIGIR 2010.
- [7] Flickr: <http://www.flickr.com/>
- [8] Google Desktop Search: <http://desktop.google.com/>
- [9] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In Y. Sure and J. Domingue, editors, The Semantic Web: Research and Applications, volume 4011 of LNAI, Heidelberg, June 2006, Springer.
- [10] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search? In Proc. of WSDM. ACM, 2008.
- [11] iPhoto: <http://www.apple.com/ilife/iphoto/>
- [12] E. Minack, R. Paiu, S. Costache, G. Demartini, J. Gaugaz, E. Ioannou, P.-A. Chirita, and W. Nejdl. Leveraging personal metadata for desktop search – the Beagle++ system. In Journal of Web Semantics, 2010.
- [13] D. Quan, D. Huynh, D. R. Karger. Haystack: A Platform for Authoring End User Semantic Web Applications. ISWC'03.
- [14] L. Sauerermann, Using Semantic Web Technologies to build a Semantic Desktop, Master's thesis, TU Vienna (2003).
- [15] J. Wang, J. Zhu. Portfolio Theory of Information Retrieval. In Proceedings of the SIGIR 2009.
- [16] Windows Desktop Search: <http://www.microsoft.com/windows/products/winfamily/desktopsearch/default.msp>
- [17] S. Xu, S. Bao, B. Fei, Z. Su, and Y. Yu. Exploring folksonomy for personalized search. In Proceedings of SIGIR, ACM, 2008.
- [18] Youtube: <http://www.youtube.com/>

Memory Support for Desktop Search

Yi Chen

Centre for Digital Video Processing
School of Computing
Dublin City University
Dublin, Ireland
ychen@computing.dcu.ie

Liadh Kelly

Centre for Digital Video Processing
School of Computing
Dublin City University
Dublin, Ireland
lkelly@computing.dcu.ie

Gareth J. F. Jones

Centre for Digital Video Processing
School of Computing
Dublin City University
Dublin, Ireland
gjones@computing.dcu.ie

ABSTRACT

The user's memory plays a very important role in desktop search. A search query with insufficiently or inaccurately recalled information may make the search dramatically less effective. In this paper, we discuss three approaches to support user's memory during desktop search. These include extended types of well remembered search options, the use of past search queries and results, and search from similar items. We will also introduce our search system which incorporates these features.

Keywords

Desktop search, memory, suggestive interface, diary study.

1. INTRODUCTION

Desktop search refers to information seeking in personal archives, which include one's emails, documents, visited web pages, digital photos, mp3 file, and mobile phone text messages. The variety and amount of items in personal archives continues to increase with the development of new computing and storage technologies. The increased complexity and size of personal archives means that more advanced desktop search techniques are needed. Since personal archive items are usually downloaded, received, created, edited or viewed (read) by the individual owning the personal archive, desktop search targets are usually what one has encountered previously. Therefore, they have some links with one's memories associated with the items. When one looks for things in one's personal archive, the approach and queries one uses may depend not only on current task context, but also rely on what one can remember about the target you are seeking. For example, with windows desktop search¹, one usually needs to recall at least one type of information about the target(s), such as the filename, or the last visiting date.

There are usually two stages for desktop search: the first is to determine what target to look for, then the second is to look for (search for) this target. For example, when I want to look for the time of a meeting later this week, I first need to know where I can find it, or where I saw such information before. After I recall that I encountered this information in an email, I will then need to recall information such as when I received this email, who sent it to me, or the subject of the email, in order to find it out

¹ <http://www.microsoft.com/windows/products/winfamily/desktopsearch/>

Copyright is held by the author / owner(s)

SIGIR'10, Workshop on Desktop Search, July 23, 2010, Geneva, Switzerland.

from my mail inbox. If I can't remember any of above information it may be very difficult for me to find this email and the information. This example indicates the important role of the data owner's and user's memory in performing re-finding tasks.

In this paper, we describe three memory feature driven approaches to assist desktop search (in sections 2, 3 and 4 respectively). In section 5 we present our desktop search system which embeds functions corresponding to these three approaches. Finally we give a brief overview of an experiment we are undertaking to test these approaches with our desktop search system.

2. USE WELL REMEMBERED INFORMATION TO SEARCH

2.1 Related Works

While it has been found that people usually prefer using simple queries and series of small steps to narrow down the dataset and approach a search target [1], entering queries to retrieve the results directly is still an important approach. There have been several studies looking at utilizing people's memory features in search (e.g. [2], [3], [4]). Most of these studies believe that the key to using memory features to support desktop search is to know which features of the items people tend to remember. Enabling users to search with likely remembered features of items is of course an important way to improve desktop search efficiency. However, this is not all. In fact there is usually another step, browsing the retrieved results to locate the target or the precise piece of information that is needed. Works looking at this aspect include [3, 5, 6], etc. In the study by Ringel et al. [6], they enabled users to browse the result on a temporal dimension together with items representing personal and public important events. They found that search times were reduced significantly when the user had access to episodic context. This implies that people's memories about their visited items (items in their personal archives) are not isolated units which are comprised only of the memory of the attributes of specific items, but rather that they are associated with the episodic context of accessing these items. In preliminary studies we also found that a subject had more reliable memory of episodic context (e.g. location) than of the target items themselves. Search queries which combined content and context information showed greater advantage over long term [7]. However, the result of this experiment was only from one subject. For this reason, we are conducting a diary study to explore what other people remember about their personal archive items when they look for them.

2.2 Diary Study

2.2.1 Participants

This is an on-going study. We have so far completed a pilot stage with four subjects. Many more participants have agreed to participate in this experiment. All the participants were university research students majoring in computer science. They were invited to participate in person. The details of the diary study were explained to them before they signed up.

2.2.2 Material

Diary books with printed questionnaires (shown in figure 1) were given to each subject. Each of the diary books contains 40 pages each of which can hold 16 diary entries. The other 8 pages include a participating consent form, instructions, and two blank pages for comments. On one side of each paper, there is a field for description of the target, one big field for free recalled details related to target, and multiple choice questions about the type of the target if the target field is not filled in because it contains private information, the frequency of access, and remembered occasions of access. On the back of each page there are multiple choices questions asking about their reasons and approaches of this re-finding activity, and a group list of episodic context features from which to select their remembered ones, such as “your location”, “people around you”, “other digital items accessed around that time”. This is because people may forget to list some information during free recall even though they remember the details of them. An electronic version of this diary book is also available online.

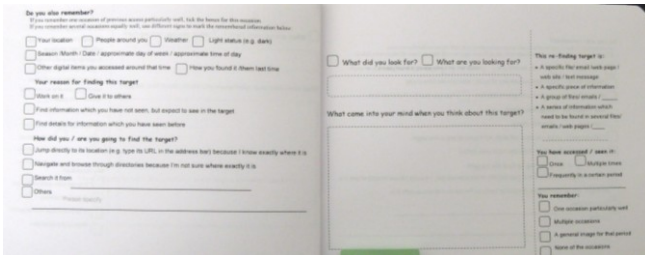


Figure 1. Diary Entry pages

2.2.3 Methods

The participants were asked to fill in a diary entry when they are looking for or after they found digital items which they have encountered before. One week after they were given the diary book, each of them was visited to interview them about the diary entries they have written by that time. This was to check if they fully understand the questions, inspire them to recall more aspects of the target related information, and encourage them to add more high standard diary entries more frequently.

2.2.4 Results so far

To date, four participants have completed the study and returned the diary book. Due to the small number of participants who have so far completed the study, statistical conclusions about remembered features cannot be provided for this paper. However, the following is a summary of our findings at this stage. For 91% of the diary entries, the participants claim to remember at least one occasion of interacting with the targets, though for 60% of these they only remember a general context. For example, one recalled “I was working on it day and night to beat the deadline”. People sometimes also remember why they

accessed that item previously, associated events or tasks, or people involved in those events. This suggests that if their remembered episodic information of interacting with the item can be recorded, it can be used in subsequent search. Another interesting finding is that they claim to remember how they found the target previously, sometimes even remembering the exact queries used to find the item. This is consistent with existing studies which show that we sometimes search for things which we have searched for before, and use similar queries [8].

2.2.5 Discussions

We believe that enabling users to search with likely remembered features of their targets and episodic features of previous access of these targets can make desktop search easier for the user. Admittedly, not all types of well remembered information can be recorded and translated to textual content to perform search. For example, what one thinks, smells cannot generally be captured. However, searching by these types of information is not impossible if past queries and results can be used given they searched by these types of information before. In the next section, we discuss a search approach using previous search queries and retrieval results.

As for the question of which episodic context information should be used, we will further explore this in our diary study and a possible follow up survey to statistically investigate the most likely remembered features.

3. USE OF PAST QUERIES AND RESULTS

3.1 Background

According to the findings of our diary study, people sometimes remember how they found information or items last time. We propose that an additional background search into a data collection of one’s past desktop search queries using the current search query can assist the user to more efficiently retrieve a currently sought item, if they remember part of a previous query and are looking for the same item this time. This is because individuals may use their remembered previous queries to search again. Although the same query may bring them exactly same result list, they may still need to browse for the exact result in the retrieved list. Therefore, each record in this data set should include not only the past query, but also the result which was found to be relevant with this query. When a matching query is found, the corresponding result (item) can be included in the result list for the current query. If this happens to be the item that the user wants, it can save them the effort to browse for it in the list. The Heystaks tool [9] can also serve this function, however it focuses on sharing past queries in a social community, which is different from personal desktop search. Besides, it is limited to only one search field. Desktop search usually has many more potential search options. Moreover, people may not always remember correctly which queries brought them the target last time if they tried several queries in rapid succession. This is because that when we recall something, we actually reconstruct the whole thing or story from atomic pieces of associated information [10]. Omitting or misplacing any pieces of memory may lead to mismatching queries. Current major desktop search tools filter the data collection and return results which match *all* search criteria. Thus, a memory mistake of a single feature can cause the search to fail. We believe that

if the search system can provide relevant items which match “any” of the multiple search criteria, such memory failure causing information retrieval problem can be reduced. This is because even if the user recalls a piece of information incorrectly for one field, the potential target may still be retrieved because it matches other criteria.

3.2 Solution

For a similar reason, people may remember how they found certain items previously, and may even remember the queries they used. However, if someone developed a query iteratively to locate the desired target, they may not necessarily remember correctly which query actually retrieved the target for them, since they generated these queries in almost the same context and the queries they used may have equal-strength links to the experience of achieving the target, if the person did not intend to learn which one is the “right” query.

Therefore, we believe that to fully utilize memory of previous search experiences, and supplement possible memory faults, not only the query which brings a “click” on the results should be recorded into the data collection of past queries, but also other queries for that same search target should be recorded together with the selected target items.

Also, if the main data collection (index of personal archives) has multiple fields and the search system supports queries of multiple fields, the index for past queries should be indexed to the corresponding fields. Similar to the situation discussed earlier when retrieving documents associated with past queries and results, items with any matched field should be considered in case the user only partially remembers the previous query.

4. SUGGEST FOGOTTEN TARGETS

The above methods focus on the second step of desktop search, that is, recalling information related to the target to search for the target. However, sometimes people do not even remember the existence of possible targets, and therefore the generated queries may not match any features of these potential targets to retrieve them. Or sometimes the image of the potential target item is too blurred and the user is unable to describe what this item is with any of the provided search options.

We believe that searching by similar items can assist people in finding these potential targets which are less well remembered or even forgotten. For the former type of potential targets which are omitted because the user forgot about its existence at the time of searching, a re-search by features from an already-found target can possibly bring up these items.

As for the later type, since it is usually much easier for familiarity-based recognition than recalling exact details, it is possible that its features can be recognized as those of a potential target if they are presented, even if they are displayed as features of some other items [11]. In other words, when the searcher sees an item which is similar to the target in certain respects, these features may be recognized as a target even if the searcher is not sure which exact features can be used to generate queries to search for the target.

4.1 Challenges brought by these approaches

While the above approaches aim to increase the chance of retrieving the correct result and improve recall, they may equally bring more noise, and reduce the precision of the search. Thus

they bring the challenge to information retrieval techniques of pushing potentially relevant targets to the top of the retrieved list. Refined search may also be needed after an initial search. The difficulty is how to support users in performing a refined search with less noise and possibly better recall, and more importantly to improve the precision. One common solution is to use filters when browsing results. However, filtering results by certain criteria may not significantly reduce the available result set. This is because people usually only select information they are certain about (possibly well remembered) as filters, so they may have used these criteria as a major part of their query too. Thus it is very likely that most of the results may satisfy these filter criteria. More efforts are needed to design browsing functions to facilitate locating information in large result sets.

5. OUR SEARCH SYSTEM

In this section we will introduce our search system which embeds the memory support functions outlined above. This system is primarily designed for accessing extended personal archive data collections. Such datasets not only include digital items that one has encountered, but also timestamps for accessing them, and the physical world context when accessing them, such as the person’s location, or the weather. With this system, we can test whether enabling search by these types of episodic context information can improve the effectiveness of desktop search for users.

5.1 IR algorithm- multiple fields

The underlying search algorithm enables search of multiple fields with “or” logic as described in section 3.1. An in-house developed implementation of BM25 [12] for Lucene was used to process these queries. Weighted individual field scores are summed to arrive at the overall retrieval score for an item. For each individual item field a ‘should occur’ query clause (see [13] for details) is used on query terms. The retrieval approach used means that only one result set will be returned for each search action. On the search interface, tags are used to enable users quickly jump to wanted search fields and remind them of possibly remembered options.

5.2 Index of past queries

To make use of our “remembered” past queries (as proposed in section 3), the system logs every search query and results selected from the search interface. A click on the search button is defined as a single search action. A search session includes all search actions for the same target(s). Queries for each search action are recorded and indexed into a Lucene index as a single document. Each document contains a field for task_id, fields for each search field, including fields for the title, keywords, item type, etc., and a field for the item_ids of targets found during the entire search session.

When a user searches, the query not only searches in the Lucene index of the desktop data collection, but also searches in the index of past queries. The results retrieved from both indexes will be presented in an integrated fashion, so that items appearing in both lists are merged to reduce the total amount of information being presented to the user. Items retrieved from the past query index will also be marked so as to give the user an indication that this item is likely to be what the user wants selected previously when entering this query. The question of

how to rank the results retrieved from the past query index is still to be explored.

5.3 Search by Similar items

This system is also designed to allow search from result items, hoping to bring more potential qualified items according to the rationale we explained in section 4. Users can click the result to look for items which are similar in content (file content, author of the email or SMS, type, etc.) or adjacent in time (as shown in Figure 2). We are also planning to enable search by physical

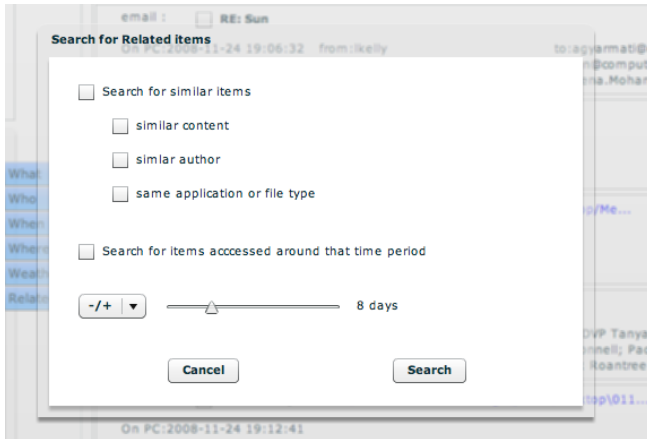


Figure 2. Screen cut of our desktop search interface: search from similar items

context of selected result items, e.g. similar weather, place, surrounding people. However, the elements which make the physical context look similar may not always be easily describable. For example, visual presentations (e.g. a photo) may be much more powerful in triggering people's memory of the physical context of the target item.

6. CONCLUSION

In this paper, we proposed three types of approaches to support the user's memory imperfectness during desktop search. Firstly, based on a review of related work, and the findings in our ongoing diary study, we suggest that episodic context can be recorded and exploited in search, and past queries can be utilized because people may remember previous queries to search again. The second approach focuses on supplementing memory by retrieving from an index of past queries to retrieve previously selected targets for the entire search session. This is because that people may not necessarily remember exactly which query brought the target to them previously. The third approach aims at overcoming memory problems at the first step of re-finding, that is, identifying the potential target. We hypothesized that search from similar items can reduce the search problems caused by memory failures at this step. We also introduced our desktop search system features which support these approaches. While all these approaches potentially enable more effective search for relevant targets, they may equally bring more noise. We are currently conducting an experiment to explore whether these approaches can improve desktop search performance, e.g. retrieve more relevant items, or reduce the time and effort spent on each search task. We expect to be able to report the results at the workshop.

7. ACKNOWLEDGMENTS

This work is supported by grant from Science Foundation Ireland Research Frontiers Programme 2006. Grant No: 06/RFP/CMS023.

8. REFERENCES

- [1] Teevan, J., *et al.*, "The perfect search engine is not enough: a study of orienteering behavior in directed search," In the Proceedings of the SIGCHI conference on Human factors in computing systems, Vienna, Austria, 2004.
- [2] Dumais, S., *et al.*, "Stuff I've seen: a system for personal information retrieval and re-use," In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, Toronto, Canada, 2003.
- [3] Elswiler, D., *et al.*, "Exploring memory in email re-finding," *ACM Trans. Inf. Syst.*, vol. 26, pp. 1-36, 2008.
- [4] Blanc-Brude, T. and Scapin, D. L., "What do people recall about their documents?: implications for desktop search tools," In Proceedings of the 12th international conference on Intelligent user interfaces, Honolulu, Hawaii, USA, 2007.
- [5] Freeman, E. and Gelernter, D., "Lifestreams: a storage model for personal data," *SIGMOD Rec.*, vol. 25, pp. 80-86, 1996.
- [6] Ringel, M., *et al.*, "Milestones in time: The value of landmarks in retrieving information from personal stores," in *Interact 2003, the Ninth IFIP TC13 International Conference on HCI*, Zurich, Switzerland, 2003, pp. 184--191.
- [7] Kelly, L., *et al.*, "A study of remembered context for information access from personal digital archives," In Proceedings of the iIIX, London, United Kingdom, 2008.
- [8] Teevan, J., *et al.*, "History repeats itself: repeat queries in Yahoo's logs," In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, Seattle, Washington, USA, 2006.
- [9] McNally, K., *et al.*, "Towards a reputation-based model of social web search," In Proceedings of the 14th international conference on Intelligent user interfaces, Hong Kong, China.
- [10] Loftus, E., "Memory Distortion and False Memory Creation," vol. 24, ed, 1996, pp. 281-295.
- [11] Cleary, A. M., "Recognition Memory, Familiarity, and Experience," *Current Directions in Psychological Science*, vol. 17, pp. 353-357, 2008.
- [12] Robertson, S., *et al.*, "Simple BM25 extension to multiple weighted fields," In Proceedings of the thirteenth ACM international conference on Information and knowledge management, Washington, D.C., USA, 2004.
- [13] Kim, J., *et al.*, "A Probabilistic Retrieval Model for Semistructured Data," in Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval, Toulouse, France, 2009.

Using Personal Information Visualization for Document Retrieval

Paulo Gomes
Instituto Superior Técnico
Av. Rovisco Pais, 49
1050-001 Lisboa, Portugal
+351-213100289
paulo.gomes@ist.utl.pt

Sandra Gama
Instituto Superior Técnico
Av. Rovisco Pais, 49
1050-001 Lisboa, Portugal
+351-213100289
sandra.gama@ist.utl.pt

Daniel Gonçalves
Instituto Superior Técnico
Av. Rovisco Pais, 49
1050-001 Lisboa, Portugal
+351-213100289
daniel.goncalves@inesc-id.pt

ABSTRACT

During our constant interaction with computers, we generate large amounts of personal information. However, it is often hard to find a certain item we are looking for, since our data is spread throughout several places and applications. Nevertheless, a meaningful visualization technique may be the solution to this problem. We present VisMe, an interactive integrated visualization system for personal information that allows users to meaningfully navigate and retrieve their data. Relevant concepts (people, subjects and documents) are uniformly displayed in interconnected timelines. Each of these items can be progressively expanded into new timelines, allowing relations between them to be explored in a simple, straightforward way. Several avenues can be simultaneously explored in context, thus giving users insights into their digital selves that current tools have a hard time providing. VisMe goes beyond traditional desktop search solutions by allowing not only keywords, but also the relations between different kinds of personal information to be used to retrieve personally relevant data.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces—*Graphical user interfaces (GUI), Interaction styles, User-centered design*; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—*Search process*

General Terms

Design, Human Factors

Keywords

Personal Information Retrieval, Personal Information Management, Information Visualization, User-Centered Design

1. INTRODUCTION

An increasing number of devices, such as laptop computers and smartphones, have pervaded our daily lives, providing us with the means to generate and store large amounts of personal information, from documents to emails. Either directly or indirectly, this data can help us understand who we are, what we do, and what we are interested in. "What was I doing in January 2005? Where can I find the article John sent me two months ago?" These are questions that an effective analysis of our personal information should be able to answer. This is not straightforward. In fact, despite considerable growth in storing capacity over the last years, methods and applications for managing and retrieving personal information have not suffered substantial improvements.

Hierarchical organization is one of the prevalent organizational systems, but it has several shortcomings. Its main disadvantage has to do with the effort requested from users to consistently classify every piece of data. Furthermore, different kinds of personal information are managed by different applications, with little or no links between themselves. A user's personal data is, thus, scattered and difficult to find.

A variety of systems have been developed to visualize different kinds of collections and histories, from emails to medical records, but while some of these systems can successfully reveal relevant patterns in the information, they are limited to particular sources. Those that are not, fail to provide a unified representation of information from different sources. We still lack an effective global visualization system for all of our personal information.

We propose a solution, VisMe, in which all relevant personal information is indexed as a whole. Links lost by applications, such as between a document in the filesystem and the email it was attached to, are recreated. Based on that index we developed a visualization centered on the most relevant autobiographic clues: time, people, and subject. VisMe allows the exploration of personal information in an efficient and understandable way. It abstracts from the different data sources to present semantically relevant information instead, and it allows several avenues to be explored simultaneously in context. As such, VisMe provides the means for finding information and retrieving interrelated items. Furthermore, by providing a synergistic visualization tool, VisMe allows users to efficiently navigate their data and helps them find patterns that are personally relevant.

2. RELATED WORK

Multiple applications have been developed to manage and retrieve personal information. *Stuff I've Seen* [3] accesses information independently of its initial form and the search is done using a word associated to the data or one of the many types of properties or metadata. The interface is halfway between browsing and keyword search. It also combines keyword and faceted search, making it easier to search for whatever criteria users do remember. *MyLifeBits* [4] stores and accesses virtually all data of a lifetime, inspired by Bush's *Memex* [1]. Given the quantity and diversity of information, *MyLifeBits* stores information and metadata.

Other solutions use information visualization as a way to search and explore personal data. Several applications have been created, usually focusing on a single information source, (email, instant messaging logs, or text documents). *Themail* [9] stands out from other email visualizations, by its simple and attractive interface and by its ability to display patterns in email content. *ChrystalChat* [8], an instant messaging visualization, displays a conversation space in an interesting three dimensional structure, even though its content representation, besides the textual display of the actual messages on demand, is limited to a peripheral mood indicator. There are also systems that allow searching and browsing to visualize information from multiple sources. *Milestones in Time* [6] takes a familiar list display and couples it with a timeline filled with landmarks to provide a simple and appealing interface for multimedia history search and browsing. *FacetMap* [7], with its facet bubbles, also avoids simple lists and manages to join a visual representation with the underlying searching mechanism in a simple and relatively effective way. *Feldspar* [2] allows users to interactively and incrementally construct association queries. Focusing more on the connections between entities and not on the entities themselves, it is possible to find things about the individuals that wouldn't be found if searching the items separately.

Evidently, none of these visualizations manages to provide a unified content overview of a heterogeneous collection of documents. This is the void *VisMe* attempts to fill: an interactive visualization of personal information taken from multiple sources which can help search for information and find relevant patterns while still allowing the micro-data (individual documents, emails, etc.) to be retrieved in context.

3. PROPOSED SOLUTION

To explore the personal document collection of a user, we first need a way to gather and index that information. *Scribe* [5], an automatic indexing application which is not the focus of this paper, is used for that purpose. It indexes and interconnects emails, documents, web pages, etc. Above this indexed data, a layer was developed to facilitate integration and to provide efficient access to the personal information. We then developed an interactive graphical user interface which handles personal information representation. We kept three goals in mind when designing this interface. First we wanted it to be simple to understand and manipulate. Second, we wanted to allow the data to be explored in context, as an inter-related whole, rather than seeing the results of individual queries one at a time. Finally, we wanted to treat information from different sources and natures in a uniform way.

The fundamental idea behind our solution is that every

element in the visualization, namely keywords (most significant words extracted from each document according to their tf-idf weight), contacts (authors, senders, or receivers of information), and documents (files, individual emails, instant messaging logs, etc.), can be expanded to display the keywords, contacts, and documents which in turn are related to them (all documents from an author, all keywords in a document, all keywords in messages from a person, etc.). Each element in the visualization is represented by a word and three buttons from which three timelines can emerge, one for each facet (Figure 1).

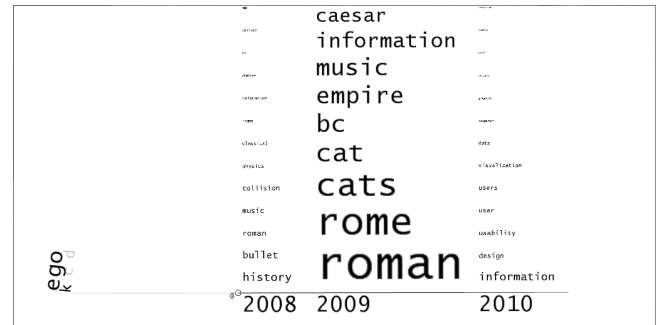


Figure 1: Expanded keywords

The visualization starts with an element representing the user ("ego"). Placing the cursor over it prompts the display of three buttons representing keywords, contacts, and documents ("k", "c", and "d"). Clicking on one of them creates a zoomable timeline on which the respective elements are displayed. The most representative elements appear larger and closer to the bottom and there is a size threshold under which elements are no longer shown. Timelines can be zoomed in progressively to display years, months, or days, by clicking on the desired period. Besides expanding a timeline perpendicularly with a simple click, users can drag the timeline out of the icon to whatever position and orientation they want. Following an expansion, any element in the timeline can be subjected to the same process as the initial element – and there is potentially no limit to this. The user is allowed a progressive exploration of their information on simple and comprehensive timelines.

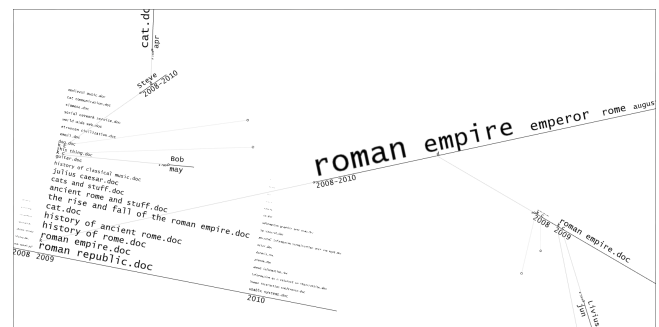


Figure 2: Timelines expanded from several elements

VisMe can maintain several expanded elements on the same canvas (Figure 2), which users can then translate, ro-

tate, and zoom in and out of using the mouse buttons and scroll wheel, so that they can observe and interconnect the various results of their ongoing research. Besides controlling the view manually, users can also double click on a timeline to bring it into focus automatically.

When users move the mouse over an element, all its instances become highlighted. This makes it easy to follow the evolution of an element over time. The color depends on the position of the mouse over the element, as there is a gradient from red (left) to blue (right). Left clicking fixes the color, another left click resets it to black.

To use VisMe as a Desktop Search tool, we provide text search capabilities. Keeping a minimalistic design, simply pressing a character on the keyboard prompts the display of an input box on the top left corner of the screen containing the text as it is written and icons ("k", "c", and "d") to define the search. As each character is written, the resulting string is searched and a possible result is shown to the user as grey text next to the string, together with an indication of the number of results for the currently selected facet (Figure 3). Pressing the tab key will complete the string to match the currently visible result, pressing the up and down keys will cycle through results.

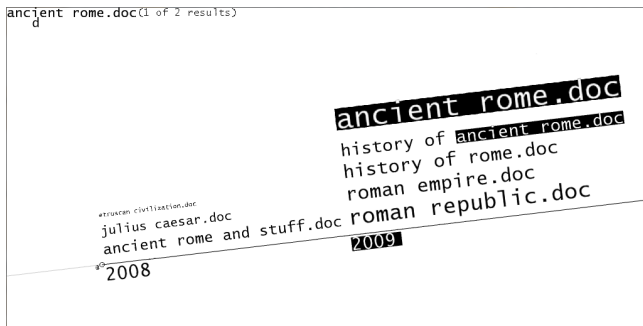


Figure 3: Searching with VisMe

If the current string appears anywhere on the expanded timelines, the respective elements will be highlighted. If there is an element that matches the search string but is not representative enough to be displayed on the timeline, it will appear on top of the respective column. It will be significantly larger than the element before it, showing that the element does not follow the same size convention as the rest of the timeline. Also, whether or not there is an exact match in any time period, the respective time will be highlighted at the bottom of the timeline.

Evidently, there is a limit to how much information can be displayed. To deal with cluttering, we implemented several measures. When the mouse is not above a timeline, only the most representative elements at the bottom are shown. Also, timelines initially appear in a horizontal state, with the most representative elements from all time periods. As such, only relatively thin lines of important elements are seen most of the time. We also let users reposition or hide existing timelines they may consider less relevant. By moving the view and reorganizing information, the necessary relevant information can be kept in view.

4. USER TESTS

Volunteers aged 17 to 29 ($\bar{x} = 23.7$, $\sigma = 2.7$) and with self reported high level of experience with computers ($\bar{x} = 3.7$, $\sigma = 0.47$, scale of 1 to 4) were asked to perform a series of tasks using the prototype application over a set of 1004 text documents authored by 102 people. This data set was crafted based on our insights on personal information spaces to be representative of a real collection of documents, with realistic trends and patterns for each tested combination of facets and enough documents and authors to make it hard to just stumble upon them without significant help from the interface. There were eight document recovery tasks, which consisted of recovering of a document with the knowledge of one or of a combination of its facets (time, most representative keyword, and author, as well as a single task in which the actual file name was given). For instance, "Find a document written by Bob about the Internet that you read in May of 2009" or just "Find a document about guitars". Before being handed the tasks, users were given a five minute demonstration of the prototype and all its features. Users were then given another five minutes to experiment with the interface freely. The order in which tasks were performed was random, in order to prevent result biases. These tasks were timed and recorded for later analysis. Users were asked to grade the difficulty of each task in a scale of 1 (very difficult) to 4 (very easy) upon completion. The time limit for each task was 150s, after which users would be told to move on to the next task. There was also a questionnaire to be answered at the end of the session which focused on the satisfaction with several general and particular aspects of the interface using a four point scale.

4.1 Results

Most tasks were completed successfully, (with failures per task in all twenty sessions $\bar{x} = 1.9$, $\sigma = 2.4$). On average, successful tasks were completed in about 52s ($\bar{x} = 52$, $\sigma = 27$). Users also considered the tasks to be easy ($\bar{x} = 3.4$, $\sigma = 0.899$). This provides some evidence that VisMe can be an effective retrieval tool for documents based on time, keywords, and authors.

There are, however, two exceptions. Both the tasks in which users were asked to locate a document based on a keyword-author combination (and, in one task, also time) have the greatest number of failures (6 and 5 in 20 sessions), the highest average completion times ($\bar{x} = 84.5$, $\sigma = 27.5$ and $\bar{x} = 88.7$, $\sigma = 27.6$) and were considered to be the most difficult tasks ($\bar{x} = 2.6$, $\sigma = 1.143$ and $\bar{x} = 2.55$, $\sigma = 0.999$). It is possible to complete these tasks, as the majority of users did, but evidently it is harder to find documents based on a combination of two facets other than time.

Completing these tasks requires either expanding documents from both facets and cross checking the results, or expanding documents from one facet and expanding the other facet out of each document one by one. Most importantly, with our data set, the number of documents that matched one of the facets (or one of the facets plus time) ranged from only two to a dozen, making it feasible to actually inspect each individual item or cross check the results of two separate timelines. More extensive result sets could have made these tasks impractical. Some users expressed some frustration with the fact that timelines were not filtered according to their hierarchy, as expanding a documents of a keyword expanded from a contact did not yield a list of documents

written only by that author about that keyword.

The questionnaire shows that users were generally satisfied with the system ($\bar{x} = 3.35$, $\sigma = 0.49$). They did not find it difficult to use for the most part ($\bar{x} = 3.20$, $\sigma = 0.69$), but they did find it somewhat difficult to learn ($\bar{x} = 2.75$, $\sigma = 0.85$). They also felt at times that the system did not offer sufficient functionalities ($\bar{x} = 2.85$, $\sigma = 0.59$). Although users did not generally consider the control over the viewing area by rotating, scaling, and translating difficult ($\bar{x} = 3.15$, $\sigma = 0.81$) we did observe that many users were uncomfortable these actions. This may be explained by the short experience with an unfamiliar and unconventional interface, but perhaps there is room for improving and smoothing out the controls, even if only for the sake of novice users.

Finally, although the retrieval times were apparently higher than those expected from keyword-based desktop search tools, such tools lack the support for certain, more complex, tasks, such as the ones required from VisMe's users. For instance, to find a document based on the combination of author, date and subject users have to perform multiple keyword queries and to interrelate information on their own, greatly increasing the overall time. VisMe provides explicit support for those tasks. Although, as described, those are the tasks where VisMe still performs comparatively worse, task completion was high, proving that it was helpful, having been designed precisely to show multiple avenues of exploration at once and to leverage the user's memory of the context surrounding each document. Future improvements to the interface (see below) will further improve its performance for these complex tasks.

5. FUTURE WORK

The tests have shown there is a problem with the retrieval of a document based on the combination of two facets. One can expect the problem to be even worse if users were to attempt a combination of several authors and keywords. A possible solution to this problem lies in filtering. We have developed a working, although still untested, solution. In the current prototype, users can simply drag any keyword, contact, and file name, into any timeline, as many times and in any combination they want, to filter it. For instance, clicking and dragging the mouse from a keyword in one timeline to the space occupied by a second timeline will add the keyword as a filter to it. Active filters appear to the left of the timeline and a simple click will remove them. This has been extended to the search string that appears on the corner of the screen, which can also be dragged into any timeline, making it easy to filter a timeline according to any existing facet that users find through textual search. We are also planning on modifying the prototype so that filters can be passed through hierarchies of timelines.

Although it was not made specifically evident in the user tests (the necessary information on screen for each task was relatively small), we have also been working on additional solutions to cluttering. The current idea being worked on is collision detection with smooth separation of timelines.

6. CONCLUSION

An efficient, integrated, visualization of personal information could allow us to search and retrieve files or discover interesting patterns. We presented our solution, VisMe, an interactive personal information visualization system. The

main idea behind VisMe is to progressively expand and lay out over time the information related to one of three facets: keywords, people, and title. It provides a unified and coherent representation of heterogeneous information and it can display an overview of the entire content of a document collection in a way that allows for the interrelation and interconnection of several of its elements. Usability tests validated VisMe's capabilities as an in-context document search and retrieval tool but also revealed complications in combining many facets in the same search, something we have attempted to solve by developing a filtering mechanism that can leverage the text search and the presence of multiple timelines on screen. Because the tests we conducted were not performed with the users' own information, they do not fully validate the solution. Ideally, VisMe takes advantage of a person's memory of their own information; the context around each document, the history of each contact, etc. The artificial data also made it impractical to compare document search and retrieval performance with traditional approaches, such as browsing document folders or searching. Still, the goal of this evaluation was to obtain an early validation of the initial progresses with our prototype, which we believe we did, and we are planning on performing more conclusive tests using actual personal information of users in the near future.

7. REFERENCES

- [1] V. Bush and J. Wang. As we may think. *Atlantic Monthly*, 176:101–108, 1945.
- [2] D. H. Chau, B. Myers, and A. Faulring. Feldspar: A system for finding information by association. In *ACM SIGCHI PIM2008, the Third International Workshop on Personal Information Management, Florence, Italy, 2008*.
- [3] E. Cutrell, S. Dumais, and R. Sarin. New directions in personal search ui. In *SIGIR 2006 Workshop, Seattle, Washington, 2006*.
- [4] J. Gemmell, G. Bell, and R. Lueder. Mylifebits: a personal database for everything. *Commun. ACM*, 49(1):88–95, 2006.
- [5] D. Gonçalves and J. Jorge. In search of personal information: narrative-based interfaces. In *IUI 2008, New York, NY, USA*, pages 179–188, 2008.
- [6] M. Ringel, E. Cutrell, S. T. Dumais, and E. Horvitz. Milestones in time: The value of landmarks in retrieving information from personal stores. In *INTERACT, 2003*.
- [7] G. Smith, M. Czerwinski, B. Meyers, D. Robbins, G. Robertson, and D. S. Tan. Facetmap: A scalable search and browse visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):797–804, 2006.
- [8] A. Tat and S. Carpendale. Crystalchat: Visualizing personal chat history. *Hawaii International Conference on System Sciences*, 3:58c, 2006.
- [9] F. Viégas, S. Golder, and J. Donath. Visualizing email content: portraying relationships from conversational histories. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 979–988, New York, NY, USA, 2006. ACM.

Towards Task-Organised Desktop Collections

Yukun Li
School of Information
Renmin University of China
liyukun@ruc.edu.cn

David Elswailer
Department of Computer
Science, University of Erlangen
david@elsweiler.co.uk

Xiaofeng Meng
School of Information
Renmin University of China
xfmeng@ruc.edu.cn

ABSTRACT

In this paper we promote the idea of automatic task-based document organisation. To make this possible we present a simplified task model and evaluate a number of algorithms for detecting which documents are associated with particular tasks. Our findings demonstrate the feasibility of such an approach, but work must be done to improve the performance for practical implementation.

1. INTRODUCTION

As people acquire ever more information as a result of personal and work activities, the management of this information becomes a serious problem and an important research issue [6]. Previous literature suggests that the tasks people perform and the activities associated with personal information plays an important role in how the information will be managed and re-found [4, 11]. There are also a number of anecdotal scenarios that highlight the importance of user task and activities in Personal Information Management (PIM) behaviour:

(1) We know that people often multi-task and experience difficulties when switching between tasks [3]. To support multi-tasking it would be useful for the user to have access to resources associated with each task; (2) When restarting a personal computer, especially after a change in workplace (i.e. from office to home) a user may need to access the files related to specific tasks in order to continue his work; (3) When starting a new task similar to a task already completed, it may be helpful to access documents associated with the previous task for reference purposes; (4) When an experienced user wants to help someone with less experience complete a task, it is often useful, for demonstration purposes, to re-find personal documents associated with the completion of this or a similar activity in the past. (5) When writing a progress report or summary of work completed, it is often useful to review completed activities retrospectively and see which documents have been created, used or modified.

To support these kinds of scenarios PIM systems need to be able to associate documents with user activities. Currently the only way to achieve this is to rely on user annotation and filing. Nevertheless, there is a large body of evidence suggesting that people are not willing or able to achieve a consistent organization, which meets all of their

needs over long-periods of time [8, 9, 13].

It would be very advantageous if tasks were able to be modelled in such a way that they could be automatically detected and appropriate documents and data items could be associated with activities. It is possible, for example, that such a model could be used to implement a task-based replacement for or enhancement to the traditional desktop metaphor. Nevertheless, there several challenges that need to be faced before such a model can be realised.

Firstly, it is difficult to define the concept of a task. Tasks can be considered at various granularities e.g. a project could be considered a task at a high-level, but would naturally consist of many sub-tasks and sub-sub tasks. Tasks can also be of various complexities [2]. Evaluating any model created or algorithm used to detect tasks is also challenging because in addition to the many problems associated with PIM evaluations, such as personalisation and privacy problems [4], there are no publicly available data sets, nor any available benchmarks or frameworks for evaluation.

In this paper we present our work in addressing these challenges. We formally define the concept of a user task and use the definition as the foundation for a task-based model for PIM. We also outline our thoughts on evaluation and describe some early results from experiments performed to test the performance of various algorithms which associate resources with tasks.

1.1 Related Work

PIM is the area of research concerned with how people store, manage and re-find information [6]. It is a multi-disciplinary field and researchers have been actively trying to understand user behaviour, such as how people interact with information and tools [9, 13], what psychological factors are important [1] and how improved tools can support user behaviour and needs [10, 5].

A large amount of PIM behaviour is performed on desktop computers, where the standard PIM model is based on the office metaphor of files and folders. Several scholars have identified the limitations of this model and suggested moving to other means of interacting with information [7, 5]. One suggested method has been to organise information based on the activities or tasks the user performs. Studies have shown the importance of activities and tasks to PIM, including behavioural strategies to allow tasks to be managed [13] and indicating that while working, people regularly need to switch between concurrent tasks [3] and have difficulty managing resources as a result. The general consensus is that the desktop metaphor provides inadequate resources for task management and switching [11] and as a result,

several prototype systems have been designed to assist with these situations, either by visualising resources in different ways e.g.[11] or making tasks the basis for organisation [12].

The common theme and, in our opinion, the major limitation of the task-based solutions proposed to date is that they all require the user to indicate which resources are associated with each task, which places the cognitive burden on the user in the same way the desktop metaphor does [8]. In this paper we explore methods to automatically associate resources with tasks. We first present a definition of a task and a model from which resources can be associated with tasks and continue propose and evaluate a number of algorithms for automatic association.

2. TASK MODEL

The basic building block for our tasks is a personal data item (PDI), which we define as *an item which has been created, accessed or modified by a person and is the result of user interaction.*

According to this definition, a file, an email or a folder can all be regarded as PDIs. What the definition also makes explicit is that PDIs only refer to items that the user has interacted with and that have not automatically been generated by software. This is important as in this work we focus on illustrating a task model and methods for identifying tasks based on desktop collections. The aim of any model would be to exploit user recollections for activities associated with PDIs and users will not remember any item with which they have had no explicit interaction.

A task has three basic elements: a goal (i.e. the thing to be accomplished), process (i.e. the activities performed to achieve the goal), and time (the period of time taken to complete the task). Each of these elements must be included and formalised in the task definition.

User activities with desktop computers can be divided into two types based on their interactions. **Read-only activities** involve only accessing a PDI, and include activities, such as reading documents, listening to music, etc.. The second type of activity, **creative activities**, involve generating new or updating existing PDIs. Thus, the goal of a creative activity can be materialized as the set of PDIs modified by the user. In this paper, our focus is on this second category of tasks. According the above criteria, a personal task can be described as follows:

DEFINITION 2.1 (PERSONAL TASK). . *A personal task PT is described as a 4-tuple (TD, DI, OL, TL) , where TD represents a written description of the task and is described as a vector of tokens. DI represents the related data items, and is denoted as a 2-tuple (GI, RI) , where GI (goal items) is a set of items generated by users during the process of completing the task, and RI (referenced items) is a set of items accessed in order to complete the task. OL (operations list) is a sequential list user operations performed to complete the task. TL describes the life-cycle of the task and is denoted as a 2-tuple (TS, TE) , where T_s is the start time of the task and T_e is the end time.*

Naturally, as mentioned above, tasks can be of varying levels of complexity. For example, notifying colleagues about an upcoming meeting would be a task requiring the creation of only one item, i.e. an email. Preparing a grant application, on the other hand, could also be considered a task, but

may involve the creation or modification of multiple items. Consequently we consider two types of tasks: *Simple Tasks*, which have a single goal item and *Complex Tasks*, which have multiple goal items.

In practical terms, a complex task can be regarded as the combination of many simple tasks. Therefore, if simple tasks can be identified then this could form the basis task identification in general. For this reason we focus here on identifying simple tasks.

3. IDENTIFYING PERSONAL TASKS

According to the definition above, there are several elements of a simple task that need to be identified: the task description, the life-cycle of the task, referenced PDIs and a task goal PDI. From this list, detecting referenced PDIs is the most challenging problem. For simple tasks, any created or modified PDI could be considered a task goal item. A task description can be generated by applying techniques, such as TF-IDF, to the PDI content or utilising a file or folder name (for non-text-based PDIs). The life-cycle information can also be easily attained by taking the created and modified times of the goal item (GI).

Below, we will outline a number of basic methods for identifying reference files based on PDI properties and patterns of user interactions. We evaluate the performance of the various methods in Section 4.

3.1 Life-cycle based method

A simple method of detecting PDIs associated with tasks is to take all files accessed within the life-cycle of a task as its references. This method will achieve perfect recall i.e. all of the files associated with a task will be detected, but the fact that people multi-task and continue tasks over long time periods will inevitably result in very low precision i.e. many inappropriate files being taken as task references. Nevertheless, the high recall property makes the approach a useful baseline algorithm for evaluations.

3.2 Directory-based method

We know from studies of folder organisations that people often organise their information items based on activity [9, 13]. Thus, a simple approach to associating PDIs with tasks is to utilise the information implicitly provided by the user through his folder structure. Given a task goal file, we can take all files located in the same folder (as well as the folder itself) as its references. An obvious limitation of the method is that files can be organised in other ways (e.g. time, people etc.) so it is likely that in some cases appropriate RIs will be located across folders – such references will not be detected using this method.

3.3 Sequential distance-based method

Another assertion we can make regarding user behaviour is that items accessed and modified within similar time periods relate to the same task. If this is true then the sequential distance – the number of items accessed or modified between two items in a sequential access list – will reflect, to some degree, the relationship between the items.

This method is simple to implement, but also has limitations. We expect it to work well when the user does not multi-task, but poorly for multi-tasking situations. Finding an appropriate threshold distance is also a problem. We assume that a small threshold will lead to higher precision

but lower recall and a larger threshold will lead to a low precision but higher recall. We provide some data regarding threshold selection in the experimental section below.

3.4 Operational pattern-based method

A further assertion we can make about user-behaviour is that after referring to a file, the user will modify the goal item of the task. If this is the case we can expect modify operations on the goal item to be surrounded by read operations of reference items for that particular goal item. Inspection of user interaction logs (see experiment below), seems evidence this assertion. We can exploit this with the following algorithm

Algorithm 1 Operational Pattern Based Algorithm

Input: An existing access list $L' = X_1, X_2, \dots, X_n$; An existing task set TS; Latest accessed file X_{n+1} ;

Output: An updated task set TS.

```

1: procedure Identify Simple Task( $L', TS, X_{n+1}$ )
2:   if then  $X_{n+1}.operation = \text{"Modify"}$ 
3:     if then  $\nexists t \in TS \wedge t.goalitem = X_{n+1}.item$ 
4:       Create a new task t
5:       Add  $X_n.item$  into t.reference
6:     else
7:       Find a task t where  $t.goalitem = X_{n+1}.item$ 
8:       Add  $X_n.item$  into t.reference
9:       Find MSL  $L'' = (X_k, X_{k+1}, \dots, X_{n+1})$ 
10:      Add  $X_i.item$  ( $k + 1 \leq i \leq n$ ) into t.references
11:     end if
12:   end if
13: end procedure

```

When a modification operation is detected, i.e. a GI is processed, the references for that task will be updated. First, the algorithm will find the latest read operation for the GI within the access list L' . Following this it checks to see if two records in L' exist, which point to the same DI. If not, the access list is denoted as a Minimum Sequential Loop (MSL) and all DIs accessed within this MSL are regarded as references of the GI. Unlike the sequential distance-based method, here no threshold value needs to be specified in advance. However, like the SD method this approach has the disadvantage that multi-tasking behaviour may negatively affect performance.

4. EVALUATING PERFORMANCE

A dataset collected via a naturalistic investigation formed the basis of our evaluation. By developing and deploying a custom-designed piece of software built based on APIs for Microsoft Windows operating system, we captured user file accesses during the course of normal PC usage. We recorded two types of operation: “read-operations” where items were accessed or read and “modify-operations”, where items were newly created or modified. For each operation we stored the associated file name, the directory-path and a timestamp.

8 participants (4 male, 4 female, aged between 25 and 40), volunteered to take part over a period of approximately one year. The participants were all researchers or research students at a major Chinese university and although they represent a relatively homogeneous population, they are all busy people who struggle with multi-tasking and PIM in

general. The data represented their activities with their office computers.

In total 54,545 operations were recorded (avg per participant = 6818 st dev = 3947). 79% of the operations were reads and 21% were modify operations.

To create a “gold-standard” from which to compare the performance of the algorithms, we conducted a second experimental phase where the same participants were asked to manually indicate the referenced items for a given set of goal items. We selected 10 goal items per participant so that those chosen included both files modified often and only a few times. To account for the difficulties in retrospectively annotating referenced files, the participants used software, which showed a goal item and a list of potential referenced items (selected by the life-cycle based method). The participants had access to meta-data about each of the items and could also open the files to examine the content before deciding if it was a referenced file. Additionally, we asked the participants to explain the relationship between the item and the goal item. We used these data as a means to evaluate the algorithms described in Section 3.

5. RESULTS AND DISCUSSION

Figure 1 depicts the performance achieved by the various algorithms. As these graphs show, it was possible to attain high recall scores – all of the algorithms achieved average scores of 0.71 and above. However, precision was harder to attain with the best performance being achieved by the OP method (0.43).

The Directory-based approach achieved the lowest recall (0.71), with some referenced files evidently being stored in different directories. Figure 1(b) shows that some participants tended to place task-related items in the same folder (e.g. user 5) and this allowed high recall to be achieved. What is also clear from the low precision scores, is that all of the participants had files in the examined directories, which they did not consider to be related by task.

Figure 1(f) shows how the sequential distance threshold (sd) influenced the performance for the SD method. Confirming our hypotheses, when the threshold=1, we achieved the optimum F-score (0.34) and precision (0.27), but as the threshold is increased, the F-score and precision deteriorated as recall improved. This result indicates that in a practical implementation of the algorithm, it would not be necessary to use a high threshold to achieve best performance.

Figure 1(a-d) shows how the algorithms performed across users, demonstrating that it was easier to achieve higher precision with some users e.g. user 6. We hypothesize that users, for which better performance could be attained, tend to multi-task less often. If this is true it highlights a paradoxical situation where the people most likely to need support are also the most difficult to provide support for. This is an obvious weakness in the methods proposed.

What our results do show, however, is that there is potential for algorithms to automatically related items by task. Our relatively simple algorithms were able to achieve good recall although improvement is needed in terms of precision. To achieve this improvement we need to investigate ways of combining evidence from the various sources exploited in the basic algorithms presented here. Even with current performance levels, we believe that, if embedded within an appropriate user interface, the algorithms could be useful in assisting users to manually organise their documents. These

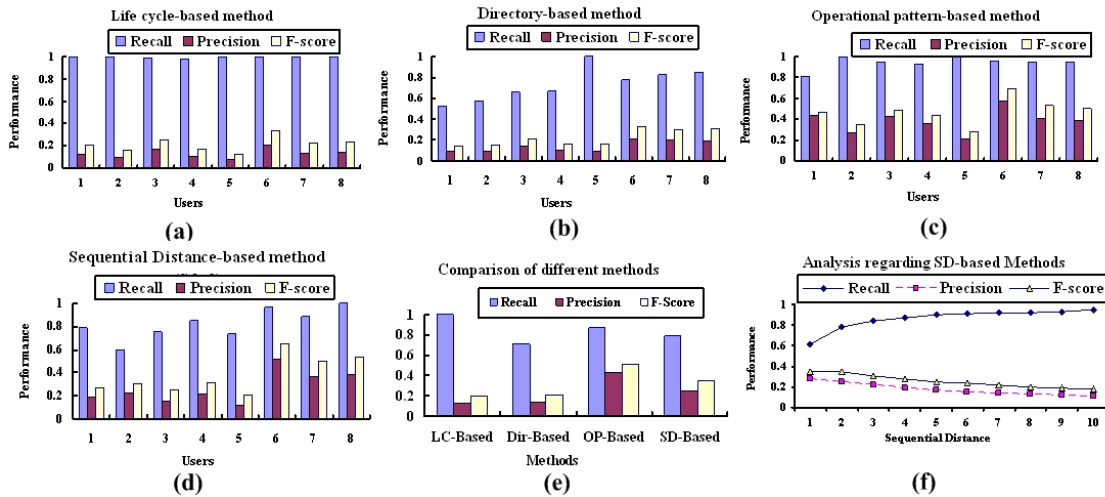


Figure 1: The Performance Achieved by the Various Algorithms

two threads represent our future plans for this work.

6. CONCLUSIONS

In this paper we have used examples from the literature to demonstrate the potential benefits of a task-oriented approach to PIM. We suggested that to realise these benefits tasks should be detected automatically and to achieve this we proposed a task model and several methods of detecting resources associated with tasks. Our model abstracts and simplifies the concept of a task, allowing the performance of the proposed algorithms to be evaluated. The evaluation results suggest that it may be feasible to achieve an automatic means of task-detection, but work must be done to improve the performance for practical implementation.

7. ACKNOWLEDGMENTS

This research was partially supported by the grants from the Natural Science Foundation of China (No.60833005); the National High-Tech Research and Development Plan of China (No.2009AA011904); and the Doctoral Fund of Ministry of Education of China (No. 200800020002).

8. REFERENCES

- [1] D. Barreau, *Special issue on the social and psychological aspects of personal information management*, Journal of Digital Information **10** (2009), no. 5.
- [2] K. Byström and K. Järvelin, *Task complexity affects information seeking and use*, Information Processing and Management **31** (1995), no. 2, 191–213.
- [3] M. Czerwinski, E. Horvitz, and S. Wilhite, *A diary study of task switching and interruptions*, CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems (New York, NY, USA), ACM Press, 2004, pp. 175–182.
- [4] D. Elswailer and I. Ruthven, *Towards task-based personal information management evaluations*, SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM Press, 2007, pp. 23–30.
- [5] E. Freeman and D. Gelernter, *Lifestreams: a storage model for personal data*, SIGMOD Record (ACM Special Interest Group on Management of Data) **25** (1996), no. 1, 80–86.
- [6] W. Jones and J. Teevan (eds.), *Personal information management*, Seattle: University of Washington Press, 2007.
- [7] V. Kaptelinin and M. Czerwinski, *Beyond the desktop metaphor designing integrated digital work environments*, MIT Press, 2007.
- [8] M.W. Lansdale, *The psychology of personal information management.*, Appl Ergon **19** (1988), no. 1, 55–66.
- [9] T. W. Malone, *How do people organize their desks?: Implications for the design of office information systems*, ACM Trans. Inf. Syst. **1** (1983), no. 1, 99–112.
- [10] G. Robertson, M. Czerwinski, K. Larson, D. C. Robbins, D. Thiel, and M. van Dantzich, *Data mountain: using spatial memory for document management*, UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology (New York, NY, USA), ACM Press, 1998, pp. 153–162.
- [11] G. Robertson, G. Smith, B. Meyers, P. Baudisch, M. Czerwinski, E. Horvitz, D. C. Robbins, and D. Tan, *Beyond the desktop metaphor*, ch. Explorations in Task Management on the Desktop, pp. 101–138, MIT-Press, 2006.
- [12] S. Stumpf and J. Herlocker, *Tasktracer: Enhancing personal information management through machine learning*, Proc. Workshop on Personal Information Management, SIGIR, 2006.
- [13] S. Whittaker and C. Sidner, *Email overload: exploring personal information management of email*, CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems (New York, NY, USA) (M. J. Tauber, ed.), ACM Press, 1996, pp. 276–283.

History Structure for Exploring Desktop Data

Harumi Murakami
Osaka City University
3-3-138, Sugimoto, Sumiyoshi, Osaka,
558-8585 Japan
harumi@media.osaka-cu.ac.jp

ABSTRACT

We present a method of data integration and associative retrieval using a simple information structure called *history structure*, which is constructed from time, keywords, and URI sets. We developed a prototype that generates a user knowledge space from various information usages (e.g., web browsing, mail, twitter, diaries, and purchases) and helps users explore their desktop data.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – *Retrieval models*.

General Terms

Design, Experimentation

Keywords

History structure, associative retrieval, knowledge space, externalized-memory model, integration

1. INTRODUCTION

We are surrounded by various kinds of information. Interest in Personal Information Management has increased in recent years [1], partly as a reaction to information overload, which is becoming a real problem in our daily lives. Much research has presented ideas for integrating such information [2, 3].

In this paper, we present a new method of data integration and associative retrieval by using a simple information structure called *history structure*, which is constructed from time, keywords, and URI sets. Our prototype generates a user knowledge space from the user's various information usages (e.g., web browsing, mail, twitter, diaries, and purchases) and helps users explore personal desktop data.

History structure is simply generated from existing information sources. Our approach resembles tagging; however, the manual tagging of personal information is time-consuming. We aim to automatically generate history structure.

First, various information is stored in the history structure. Next, the system creates knowledge spaces for individuals. Figure 1 displays an overview of our approach.

Our approach, which is based on an externalized-memory model inspired by a human memory model, is presented in Section 2.

Copyright is held by the author/owner(s).

SIGIR'10, Workshop on Desktop Search, July 23, 2010, Geneva, Switzerland.

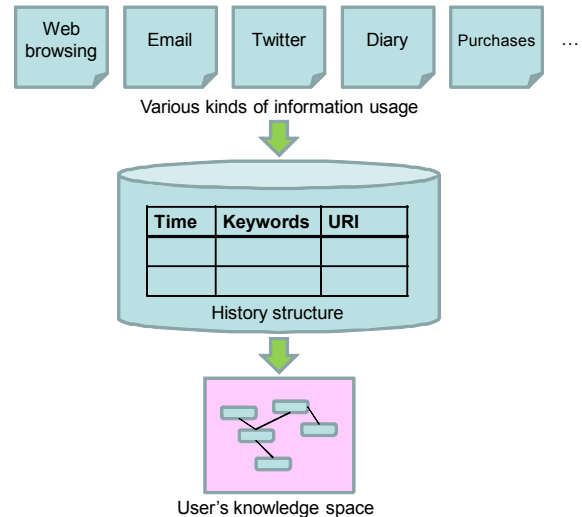


Figure 1. Overview

2. EXTERNALIZED-MEMORY MODEL

Externalized-memory is a concept that virtually externalizes and stores the contents of human working memory on computers. We capture information processed during human cognitive processes by working memory. History structure is a kind of externalized memory.

Figure 2 shows an overview of our proposed model. Such information processed by the user as browsing the web is accumulated into the user's externalized memory (history structure) by a sensory memory and working memory. The knowledge space is a simulated visualization of the history structure like semantic networks that are generated from the history structure. When a user selects a keyword in the knowledge-space browser, the system searches through the history structure, recalls related information, and displays it in the knowledge-space browser.

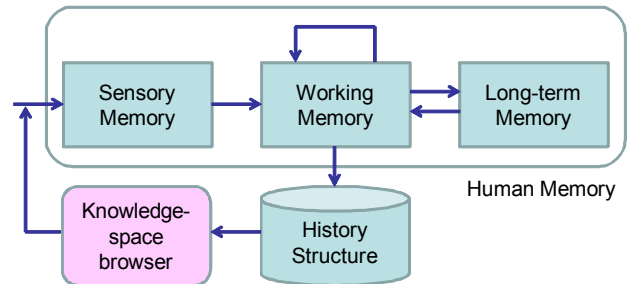


Figure 2. Externalized-memory model

Improving Re-Finding upon Work Resumption

Thorsten Prante
Duisburg-Essen Univ. (ext.)
Liebfrauenstr. 95
64289 Darmstadt, Germany
+49(0)179 108 1679
thorsten@prante.eu

Jens Sauer
Darmstadt Univ. of Technology (stud.)
Hohemarkstr. 117
61440 Oberursel, Germany
+49(0)176 2077 2074
iri3@gmx.net

Albrecht Schmidt
Duisburg-Essen University
Schützenbahn 70
45117 Essen, Germany
+49(0)179 108 9684
albrecht.schmidt@acm.org

ABSTRACT

This work presents an approach to reducing user efforts when re-orienting to an interrupted task. We present a time-centric Journal view of the user's information-work activities, which allows going back to 'task states' for activity resumption. We further compute an activity-induced correlation function to reflect information coherence, as experienced on the user's side when 'using information items together'. We thereby investigate the expressiveness of easy to understand correlation indicators, such as temporal proximity, window switching, and clipboard use. The work most closely to ours is [2],[4].

Categories and Subject Descriptors

H.5.2 [User Interfaces]

TASK SWITCHING & INTERRUPTIONS

One of the everyday experiences of information workers is that task work must be interrupted and later resumed. This particularly applies to *complex tasks* such as authoring an overview report or preparing a presentation. The corresponding *activities* can only seldom be completed within an uninterrupted period of work and hence often extend over several days: "Complex, 'returned-to' tasks comprise a significant portion of an information worker's week, but reacquiring such tasks is considered difficult by users." [3]. The basic prototypical case of *task switching* is a user interrupting her primary task for a secondary one and later resuming the primary task.

One of the most often reported effects of information workers being interrupted in their work, is them losing track of their position in the work process. An interruption necessitates a user to leave her immersion in task work and to redirect her focus of attention to the interruption source. Especially, in the case of *unanticipated* and more or less *abrupt interruptions* ([1],[3]: 20%), this can cause *forgetting information*, which had just before been under conscious control. Forgotten information can mean (at least temporarily) not remembering a) *what* to continue, b) *where* to continue. Both phenomena possibly necessitate redundant activities and hence additional "cost".

RESUMING WORK

As compared to generally orienting oneself about what to do next, the case of resuming one of several active tasks requires an in-

Copyright is held by the author/owner(s).

SIGIR'10, Workshop on Desktop Search, July 23, 2010, Geneva, Switzerland.

formation worker to remember what she did and what thereof is still unfinished – together matching (a). Having decided, which task to take up, she now needs to *re-find* the hooks, i.e. information items for continuing to work – matching (b).

These items serve as *reminders* as well as *connection points* for the to-be continued activities. As the information relevant for processing complex tasks are commonly distributed over multiple information items, the activities to advance a task often entail juggling with several items at once, i.e. during an *activity phase*. Consequently, there might be multiple items relevant for resuming work. Accordingly, we speak of *connection sets*, where each information item might have a specific reminder function.

One approach for determining connection sets, is re-finding those information items, which have been the object/s of a user's activities by the time when the interruption occurred (*activity-phase specific approach*).

REFLECTION JOURNAL

The ReflAction Journal (RJ) *visualization* is based on computer-observable fractions of information work. In fig.1, the doing pane shows *item-related user activities (uA)*, which mirror that a user's activities leave traces on those items. Accordingly, an *activity object* is composed of an information item (defining its name), and the traces related to the item, where the latter are sets of {focus, edit/view, visibility} intervals, representing, if so, interrupted user activities.

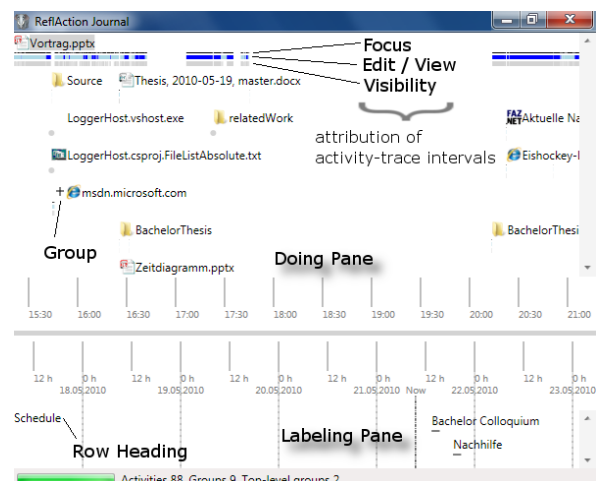


Figure 1. ReflAction Journal: The Doing-Pane view is rather zoomed-in, where the Labeling Pane [5] provides overview.

Overall, the RJ provides a representation “from which the user can determine what she had been working on (...) which was shown to strongly support resumption” [7].

The RJ *user interface* allows continuous scrolling and *zooming*, where zooming primarily affects the resolution of the x-axis. The information items shown can be accessed, if available.

For *re-finding an interrupted activity phase*, represented by a collection of activity objects in the RJ, the following ‘index notions’ can be employed for orienteering [6]: 1) *Absolute* time, 2) (name of) any information item involved in the activity phase just before interruption, 3) items at which activity was directed *before* or *after* the interruption. Additionally, 4) the *attributions* of the trace intervals facilitate ‘queries’ like “I remember having looked at those pictures I want to re-access now, just before answering (and therefore viewing) this Email I have at hand now.” These re-finding means improve the common user experience that computers do not well support the “*leaving the task (state) as it is*” strategy [1], as these means are not hampered by closed windows.

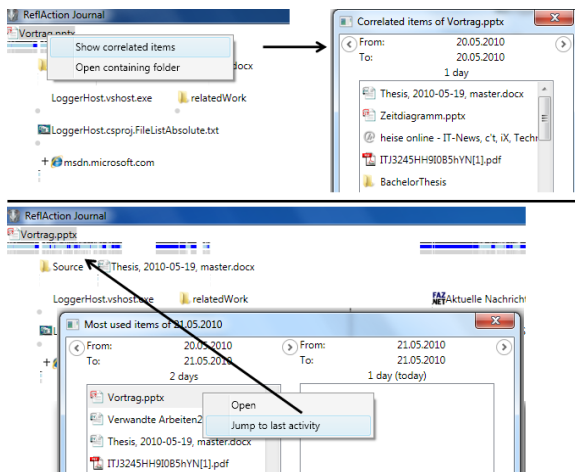


Figure 2. Employing ReflAction Correlated to view items which have been “used together” across activity phases & ReflAction Most-Used to jump to an activity (phase).

CORRELATED & MOST-USED ITEMS

Consider that a few days have passed since a user extensively used some important information sources S_i of a presentation (or any other) document. Accordingly she still *perceives the S_i as related to her presentation document a* . Therefore, we provide a *uA-related* function, which is used to determine correlated items within a time frame. Its user interface (fig.2) is an early prototype and doesn’t allow convenient control over time scoping yet.

The intuition behind this function is that information items ‘used together’ implicitly gain *activity-induced* coherence. Let $uA-related(a, b)$ denote the function mapping (a, b) to the correlation strength of b to a . The construction of this function is illustrated in fig.3. It is called for all b ’s within the provided time frame. So, $uA-related(a)$ for ‘today’ yields the list of items presented in fig.2.

So far, we only tested $uA-related(a)$ for time frames of a few days and the results are comprehensible. All variables are still subject to experimenting.

The *most-used list* also shown in fig.2 provides the main working document to a user. It computes the aggregated length of activity

intervals per item and time frame, where editing has more weight than viewing.

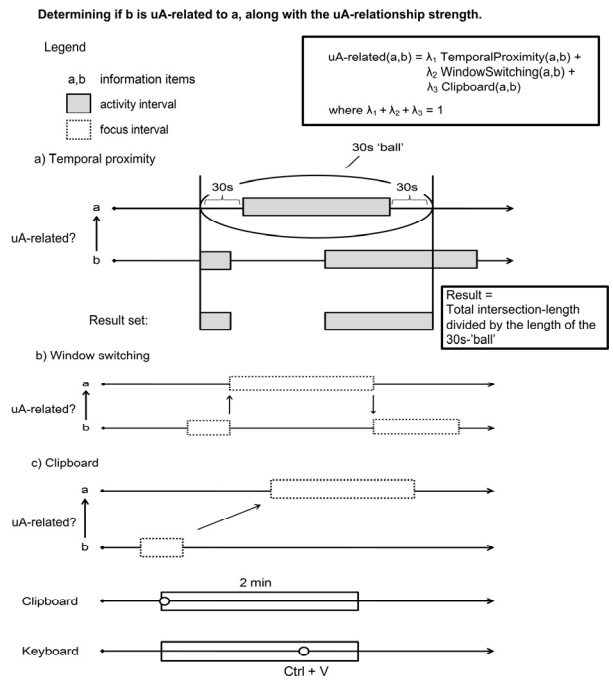


Figure 3. Construction of the *uA-related(a, b)* function from activity-oriented indicators: Window Switching and Clipboard Use implement a counting of the corresponding events.

CONCLUSIONS & FUTURE WORK

Our activity representation fed by the ContextDrive [5] system can be considered *personal metadata*, which allows expressing user-experienced correlations also among non-text information types. While our current main task is experimenting with the here presented indicators and tuning their weights, we will also work on including non-text information types, which are currently under-represented. This will allow us to better evaluate and compare our work to other means representing relationships among information items, such as semantic technologies.

REFERENCES

- [1] Bondarenko, O. (2006). Task Switching in Detail. *Proc. DCEIS'06*, Doctoral Consortium.
- [2] Brdiczka, O. (2010). From Documents to Tasks: Deriving User Tasks from Document Usage Patterns. *Proc. IUI'10*.
- [3] Czerwinski, M., Horvitz, E., & Wilhite, S. (2004). A diary study of task switching and interruptions. *Proc. CHI'04*.
- [4] Pedersen, E. R., & McDonald, D. W. (2008). Relating Documents via User Activity: The Missing Link. *Proc. IUI'08*.
- [5] Prante, T., Sauer, J., Lotfy, S., & Schmidt, S. (2010). Personal Experience Trace. *CHI'10 Workshop on Pers. Inf.*
- [6] Teevan, J., et al. (2007). How people find personal information. In: Jones, W. & Teevan, J. (Eds.), *Personal Information Management*. University of Washington Press.
- [7] Trafton, J. G., & Monk, C. A. (2007). Task Interruptions. In *Reviews of Human Factors and Ergonomics*, Vol.3..

Can Maps Provide the Answer to Desktop “Search”?

Roy A. Ruddle
University of Leeds
Leeds
UK

+44 (0)113 343 5430

r.a.ruddle@leeds.ac.uk

ABSTRACT

During their lifetime, an individual may accumulate millions of items in their personal information space. This article outlines how maps should be designed to help individuals to find items in a large information space. Recommendations are made for the layout and visual encoding of information space maps, and how such maps could present search results and allow rapid browsing within a single, integrated view.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation (e.g., HCI)]: User Interfaces – *graphical user interfaces (GUI)*. H.5.4 [Information Interfaces and Presentation (e.g., HCI)]: Hypertext/Hypermedia – navigation. H.3.3 [Information Storage and Retrieval]: Information Search and retrieval – search process.

General Terms

Design, Human Factors.

Keywords

Maps, visual encoding, personal information space.

1. INTRODUCTION

An individual’s personal collection contains many types of information, from documents, pictures and emails, to details of web pages the person has visited (e.g., a URL). The quantity of each type of information depends on the individual, but varies by several orders of magnitude. For example, during a lifetime an individual may create 10^3 documents and take 10^5 photographs, but receive 10^6 emails and visit 10^6 webpages.

Even if all this information is integrated so that it may be accessed seamlessly from a single device, finding a particular item can be very difficult. The present article recommends how maps should be designed, to harness our natural spatial ability and the power of advanced search technology, and help individuals to find items in

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference’10, Month 1–2, 2010, City, State, Country.
Copyright 2010 ACM 1-58113-000-0/00/0010...\$10.00.

personal information spaces.

2. ORGANIZATION AND ACCESS

To achieve broad acceptance, any personal information retrieval interface needs to support individuals’ preferences for organizing and accessing information. The three main methods people use to organize information are: (a) don’t (characterized by there being a very large number of files or emails in one folder), (b) hierarchical, or (c) tags. The main mechanisms for accessing information are: (i) browse, (ii) search, and (iii) scan. The distinction between browsing and scanning is important because browsing refers to a large-scale space, which a person navigates through until the desired item of information is found, whereas scanning involves a small-scale space (e.g., a list of search results) that is inspected to identify the item that is being sought. In the real world, large- and small-scale spaces (e.g., a building vs. a room) involve fundamentally different cognitive processes [15].

Table 1. The main methods people use to organize and access information in a personal collection.

Organization	Access method		
	Browse	Search	Scan
Don’t (flat structure)		*	*
Hierarchical	*	*	
Tags	*	*	*

The relationship between different organization and access methods is summarized in Table 1. By definition, if information is in a flat structure then browsing is not possible. Instead, a person either uses a search engine to reduce the items of information to a set of possibles and then scans to identify the item that is sought, or just scans a list of the items. In both cases, ordering the information (e.g., by name, date or file type) reduces the number of items that need to be scanned. Hierarchical structures are designed to be browsed, and rely on the person recalling or recognizing the correct choice as they navigate from one level of the hierarchy to the next. By contrast, use of search bypasses the hierarchy and presents the person with a set of possibles that need to be scanned. The only link between browse and search displays are the names of files and their paths. Tagged information may be browsed, searched or scanned, depending on how an interface exploits the structure provided by the tags.

There is clear evidence that people prefer to browse to find information within a file system, and only use search as a last resort [2]. However, no one has yet developed a way of portraying information spaces in a map-like form that aids information retrieval.

3. MAP DESIGN RECOMMENDATIONS

Humans are proficient navigators. We quickly learn the layout of new environments (e.g., a holiday resort) and, when provided with a navigational aid such as a map, are very adept at finding places we have never visited before.

Information spaces (desktop file systems, websites, etc.) are harder to navigate than the real world, despite the fact that the number of items in an information space is typically smaller than the number of buildings in a large city. Therefore, if an effective way of mapping information spaces can be developed then those spaces should become as navigable as the everyday environments in which we live and work.

3.1 General Benefits of a Map

There are a number of general reasons why information spaces are difficult to navigate. These are outlined below, together with the ways in which maps can help:

- a) *Sensory information is impoverished.* Most real world settings contain rich visual detail, which provides a multitude of cues to aid navigation. By contrast, each view of a given information space (e.g., the contents of a desktop folder) has a similar look and feel. This could be addressed by annotating each folder's region on a map with a distinctive landmark, echoing the aesthetic properties of many historic maps.
- b) *Only local spatial information is provided.* During desktop browsing, the contents in one folder are replaced on the screen by the contents in the next folder that is selected. A map addresses this by placing each folder in a specific position, so an individual's spatial memory for that folder is reinforced every time it is viewed.
- c) *Information space navigation is like traveling in thick fog.* In the real world, extended lines of sight (e.g., down a street, across many intersections) greatly facilitate the process of navigation [7; 11], whereas with browsing windows you can only see one step ahead (e.g., the contents of your current folder). A map complements this by also displaying your surroundings.
- d) *Search and browse use entirely separate interfaces.* However, maps could be used to present search results and allow browsing within a single, integrated view. This should improve individuals' memory for the location of items within a personal information space.

3.2 Map Layout Recommendations

This section recommends how a map of a personal information space should be laid out. Key factors are how should the information be organized, level of detail, and how should the dynamic nature of an information collection be handled?

Most individuals organize their information to a certain degree (e.g., a desktop folder structure). However, these structures tend to be flat (broad, but not deep) to reduce the number of clicks (hence cost) that are required to browse to a given item of information. A fundamental advantage of maps is that they allow direct access to any point within a space, so a deeper hierarchy is

likely to be beneficial, but if the map allows direct manipulation of the hierarchy then the choice may be left to each individual user.

Recommendation: Make the map editable, so the user can define the structural layout.

Automated algorithms need to be developed to calculate a map's spatial layout from its structure, for which there are a number of approaches. Traditional tree diagrams are not suited to information spaces because the large aspect ratio makes poor use of display real estate, and folding is not desirable because of the large amount of user interaction that is required. Radial diagrams (e.g., [5]) have a better aspect ratio (closer to 1:1) than traditional trees, but are not readily comprehended by the general public. Astronomical (starfield) approaches generate sparse layouts that make inefficient use of real estate [6], although the simplicity with which regions can be defined by drawing a constellation boundary could be incorporated within other approaches. The most compact layouts use a space-filling approach, with a tree map [12] being the best known automated algorithm and hand-drawn Z-diagrams [8] have been proven to help people design information spaces.

Recommendation: Use a space-filling spatial layout.

Broadly speaking, people find information in two stages [14]. A global stage involves finding the item's locality (e.g., browsing through a space to the region that contains the item information, or issuing a suitable search query), and a local stage involves finding the item itself (e.g., browse locally, or scan the search results). A map supports the global stage, but research is required to determine the level of detail that the map needs to show. Studies of web navigation [10] suggest that the map needs to support navigation to within one or two steps of the desired item, which implies that most (if not all) folders need to be shown. Individual files could be displayed in a secondary, local view that fulfills the role of current file managers (e.g., Windows Explorer).

Recommendation: Map shows a folder level of detail, which is complemented by a local, file level of detail.

Research conducted using simulators, shows that maps oriented to match your view of the world ("forward-up") are best for deciding where next to travel, but maps with a constant orientation ("north-up") are best for learning the overall layout of a space. However, the addition of a simple you-are-here marker, which shows your momentary position and orientation, produces a map that is near-optimal for both types of task [1]. Applied to information spaces, this means that every item should reside in a fixed position, so a user may always find it by looking at the same place on the map. The only exception would be if the user deliberately moved the item within the file system.

Recommendation: Use a world-referenced ("north-up") layout.

If every item resides in a fixed position, how should the map initially be organized so that it can accommodate changes in the information a person stores over their lifetime? Research is needed to characterize the rate and magnitude of these changes, and knowledge techniques from dynamic graph drawing [4] may prove useful to generate stable spatial layouts.

Recommendation: Further research required into stable layouts.

3.3 Visual Encoding

Visual encoding refers to the graphical attributes (e.g., position, size, shape, orientation, hue, brightness and texture) that dictate how a given piece of information is rendered onto a display [13]. Cartography, the longest established field of visualization, exploits combinations of these graphical attributes to display a wide variety of information in an easy to comprehend form. Careful overloading of the attributes means that categories of information often differ in at least two dimensions (e.g., line color and width for roads; see Figure 1), so attributes may be reused for different categories of information (e.g., red lines for roads and paths).

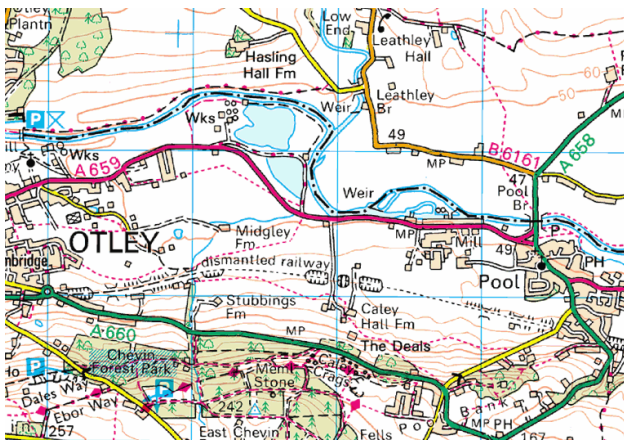


Figure 1. Ordnance Survey 1:50000 map, using color to code classes of road (red, green, orange and yellow lines, of different widths) and types of path (dashed red lines, with differently shaped symbols), and different symbols to identify the types of tree in a wood. © Crown Copyright ED 100018888.

All “maps” of information spaces use position to encode the relationship (e.g., hierarchical or topic) between items of information. Most maps only use two dimensions, but some Z-diagram examples successfully exploit the third dimension to distinguish different levels in an information hierarchy and reinforce this using color (e.g., the Nature Neuroscience website design [8]).

In most cases, however, color is used in an ill thought out manner, with all three perceptual dimensions (hue, value & chroma) varied simultaneously and colors allocated simply so that neighboring items are different (e.g., [3; 9]). The result is a kaleidoscope effect, which implies relationships between items of information that are actually wholly independent. More appropriate use of color has been made to indicate the density of documents in a topic and search query relevance [16], and standard desktop interfaces (e.g., Windows Explorer) successfully combine color with shape to create salient icons that indicate a file’s type (e.g., Word vs. PDF).

A major challenge is to develop a visual encoding scheme for information spaces, which is as rich and clear as the schemes used in cartography. Clearly this requires substantial further research, but we make the following initial *recommendations*:

- *Position*: 3D coding of hierarchy.

- *Size*: Object volume on map indicates size of items in a folder, which will allow large items to both stand out and act as reference points for remembering the location of other items.
- *Color*: Separate usage of each perceptual dimension. One possibility would be chroma and hue for depth and branch of a hierarchy, respectively.

To create a map that allows integrated browsing and searching, the following additional information could be encoded. User’s main browsing main paths and frequently accessed items should be displayed to provide a framework for spatial memory, mimicking the role of paths and landmarks in real-world navigation. However, unlike the real world or existing file browsers, the map would allow users to access any item with a minimal number of clicks, thereby saving a substantial amount of time during each working day. On the occasions when users still needed to search, a traditional textual results list could be presented in the context of the map (e.g., marking the location of each result and graphically encoding the rank). This would allow greater flexibility for results presentation (e.g., category-based ranking) and improve users’ knowledge of the location of items so more could be accessed rapidly by browsing.

4. SUMMARY

This article outlines how information space maps could be encoded, to integrate browsing and searching and allow faster access to items stored in a personal information space. Our vision is that today’s file managers will be replaced by a map-like interface that saves time every day for millions of computer users. However, major challenges remain to determine how information should be encoded on the maps and to develop algorithms that automate the process of map generation.

5. REFERENCES

- [1] Aretz, A. 1991. The design of electronic map displays. *Human Factors* 33, 85-101.
- [2] Bergman, O., Beyth-Marom, R., Nachmias, R., Gradovitch, N. and Whittaker, S. 2008. Improved search engines and navigation preference in personal information management. *ACM Transactions on Information Systems* 26, 1-24.
- [3] Chen, H., Houston, A. L., Sewell, R. R. and Schatz, B. R. 1998. Internet browsing and searching: User evaluations of category map and concept space techniques. *Journal of the American Society for Information Sciences* 49, 582-603.
- [4] Frishman, Y. and Tal, A. 2008. Online dynamic graph drawing. *IEEE Transactions on Visualization and Computer Graphics* 14, 727-740.
- [5] Grivet, S., Auber, D., Domenger, J. P. and Melancon, G. 2006. Bubble Tree drawing algorithm. In *Computer Vision and Graphics*, 633-641.
- [6] Hearst, M. A. 2009. *Search user interfaces*. Cambridge University, Cambridge, UK.
- [7] Hillier, B., Penn, A., Hanson, J., Grajewski, T. and Xu, J. 1993. Natural movement: or configuration and attraction in urban pedestrian movement. *Environment and Planning B: Planning and Design* 20, 29-66.
- [8] Kahn, P. and Lenk, K. 2001. *Mapping web sites*. Rotovision, Hove, UK.

- [9] Rivadeneira, W. and Bederson, B. B. 2003. A study of search result clustering interfaces: Comparing textual and zoomable user interfaces. In *Technical Report HCIL-2003-36, CS-TR-4682* University of Maryland.
- [10] Ruddle, R. A. 2009. How do people find information on a familiar website? In *Proceedings of the 23rd BCS Conference on Human-Computer Interaction (HCI'09)*, 262-268.
- [11] Ruddle, R. A. and Péruch, P. 2004. Effects of proprioceptive feedback and environmental characteristics on spatial learning in virtual environments. *International Journal of Human-Computer Studies* 60, 299-326.
- [12] Shneiderman, B. 1992. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics* 11, 92-99.
- [13] Spence, R. 2007. *Information visualization: Design for interaction*. Prentice Hall, Upper Saddle River, NJ.
- [14] Teevan, J., Alvarado, C., Ackerman, M. and Karger, D. 2004. The perfect search engine is not enough: A study of orienteering behavior in directed search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* ACM, New York, 415-422.
- [15] Weatherford, D. L. 1985. Representing and manipulating spatial information from different environments: Models to neighborhoods. In *The development of spatial cognition* R. Cohen Ed. Erlbaum, Hillsdale, New Jersey, 41-70.
- [16] Whiting, M. A. and Cramer, N. 2002. WebTheme: Understanding web information through visual analytics. In *Proceedings of the First International Semantic Web Conference (ISWC'02)* Springer-Verlag, London, 460-468.

Metasearch tools for desktop search

Paul Thomas
CSIRO
Canberra, Australia
paul.thomas@csiro.au

David Hawking
Funnelback Pty Ltd
Canberra, Australia
david.hawking@acm.org

ABSTRACT

Desktop search, as commonly implemented, is not capable of searching many of the web sites and other sources people use day-to-day. A metasearch model, incorporating desktop search tools and others into a “single box”, offers a potential solution. We describe this model and introduce PERS, a library for building desktop metasearch software.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; H.3.4 [Information Storage and Retrieval]: Systems and Software—*distributed systems*

1. METASEARCH ON THE DESKTOP

Desktop search is, at least in part, motivated by a desire to provide a single point of access to all of an individual’s information. This information could come from any or all of a variety of sources (Figure 1).

As normally implemented, desktop search relies on a local index. Some sources, such as local files, contact lists, or calendars, are little trouble in this model, but other sources pose a greater challenge to desktop search: enterprise resources such as proprietary databases, intranet pages, or LDAP directories require adapters or special processing. Outsourced enterprise applications may be accessible only via limited APIs. The public web, including both pages already visited (and in the browser’s cache) and pages not yet seen, is of course too large to process at the desktop. And most troublesome are the large number of online services which are private, are not crawlable, or which charge fees for access, and are therefore not indexed by public search engines. These vary from online catalogues, which are not crawlable, to large fee-for-use databases. Even if there were adapters for every collection, the size of the collection or the rules surrounding access would make local indexing impossible in many cases.

Since it is not feasible—or even possible—to index some sources locally, this precludes access to these sources from

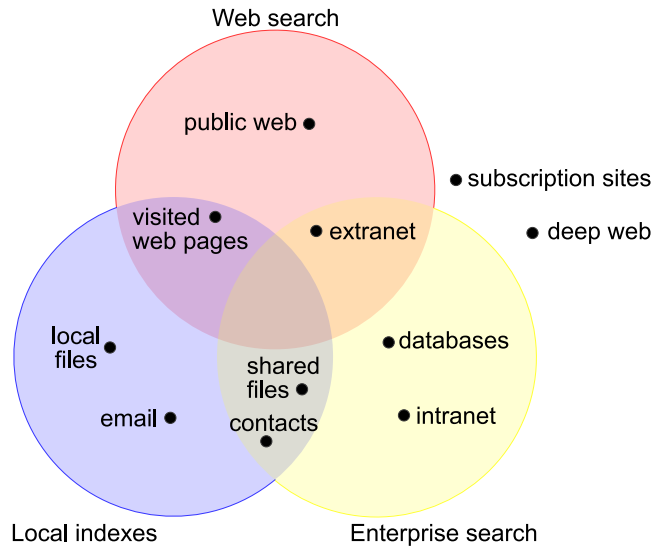


Figure 1: Sources we may wish to search, and approximate coverage of various search technologies. “Deep” web sites and subscription sites are not searchable by any of the three technologies here, although they may present their own search interface.

such local-index tools. Similarly, web search engines are limited to the public web and exclude local or enterprise content; and enterprise search engines exclude local data or the public web. Some desktop search products do offer search of the public web, either through affiliation with a web search engine or in a very limited sense through a search of visited or cached web pages. However, none of the three search technologies cover outsourced enterprise applications, subscription services, web sites which require some sort of credential (e.g. usernames and passwords), or the “deep web”.

Without a single search interface to all of a person’s information resources, the amount of time taken to find things increases at least linearly with the number of search interfaces used. For example, finding comprehensive background on a significant customer may require separate searches of an employee’s desktop, the corporate email archive, externally hosted sales database, subscription financial health reports, and the whole web. There is also a greater chance of missing something useful or important: if a choice of search interfaces is offered, users may simply search in the wrong place.

Metasearch, also known as “distributed” or “federated”

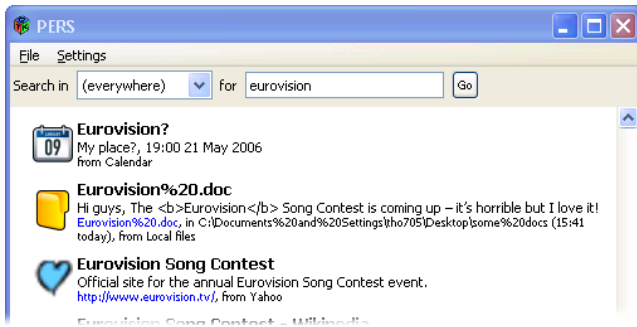


Figure 2: The PERS desktop interface. Shown are partial results from a query covering a calendar, a local file collection, and the public web.

search, offers a way to provide comprehensive, single-search-interface search. Rather than integrating at the data level by building a local index, it integrates the results of multiple search engines, each with their own index.

Desktop metasearch brings its own challenges and opportunities, and it is not yet clear how many lessons from other metasearch applications will carry over. Can a system automatically discover which search engines should be integrated? Can it characterise those engines well enough to enable intelligent selection which subset of available search engines should be consulted for each particular query? Does the query need to be translated prior to transmission to particular engines? Finally, how can the results sets from multiple engines be merged into a single high quality set and presented in a useful way?

In general the metasearch model is unable to rely on any form of cooperation from the underlying search engines, other than returning a result set in response to a query. However, an advantage of the approach is that the metasearcher can rely on the underlying search engines to deal with problems such as indexing; access control; thesauri and query expansion; and ranking.

Uncooperative metasearch has been well studied within artificial test collections [1, 3] but not yet in the desktop context. This is perhaps surprising as desktop search may be seen as a “killer app” for metasearch, but is largely because of difficulties in conducting experiments in this more realistic setting. The PERS metasearch library may overcome this difficulty.

2. PERS, A METASEARCH LIBRARY

PERS (<http://es.csiro.au/pers/>) is a *personal* search library for supporting and evaluating metasearch approaches to desktop search. PERS can handle a wide variety of data types and scales: prototypes have covered collections ranging from small personal calendars, through library catalogues, to the entire public web. It implements a “hybrid” model [2], meaning it can use a combination of existing search services and its own local indexes to search both local files and data on any subscription, corporate, or public service. A variety of merging and presentation options means it can present results from all sources in a single interface (Figure 2).

The PERS library comprises many components for metasearch at the desktop, including:

- A variety of configurable routines for “uncooperative” metasearch. This includes characterising search engines (sampling and estimating size); building language models; selecting sources for a given query; carrying out searches in parallel; and merging results.
- Search for sources including local files, calendars, addressbooks, local and remote email, the public web, particular websites, and anything with a web interface (which includes most subscription services).
- Filters for a number of file formats including HTML, PDF, and Microsoft Office.
- GUI and web front-ends.
- Support for experimentation and a range of utility functions: alternative algorithms can be switched in or out (often with a single line of code), execution can be logged, interface actions can be recorded and reported to a central host, etc.

PERS is implemented in C# and has been run under Windows, Linux, and OS X. Simple uses of the library need only a dozen lines of code.

It is easy to add other routines, other filters, and to harness other search engines. To date PERS has supported experiments in evaluation, interface design, and server selection, in web and desktop search contexts [e.g. 4, 5].

3. CONCLUSION

There is some indication from brochure sites that leading commercial desktop systems may be taking tentative steps down the metasearch path. However, this path is still very much untrodden. Clearly research is needed, and we hope that the PERS library will enable and encourage academic researchers to work on these very interesting problems. Such research can be undertaken without the need for expensive infrastructure or access to massive data held by public search engines. PERS is potentially a valuable way of studying aspects of searcher behaviour, in the context of their everyday search tasks.

4. REFERENCES

- [1] J. Callan. Distributed information retrieval. In W. B. Croft, editor, *Advances in information retrieval*, volume 7 of *The information retrieval series*, pages 127–150. Springer, 2000.
- [2] N. Craswell, F. Crimmins, D. Hawking, and A. Moffat. Performance and cost tradeoffs in web search. In *Proc. Australasian Document Computing Symposium*, pages 161–170, 2004.
- [3] W. Meng, C. Yu, and K.-L. Liu. Building efficient and effective metasearch engines. *ACM Computing Surveys*, 34(1):48–89, Mar. 2002.
- [4] P. Thomas and D. Hawking. Experiences evaluating personal metasearch. In *Proc. IiX Symposium on Information Interaction in Context*, pages 136–138, 2008.
- [5] P. Thomas, K. Noack, and C. Paris. Evaluating interfaces for government metasearch. In *Proc. IiX Symposium on Information Interaction in Context*, 2010. To appear.