iMecho: An Associative Memory Based Desktop Search System

Jidong Chen, Hang Guo
EMC Research China, Beijing, China
{chen_jidong,guo_hang}@emc.com

Wentao Wu, Wei Wang
Fudan University, Shanghai, China
{wentaowu,weiwang1}@fudan.edu.cn

ABSTRACT

Traditional desktop search engines only support keyword based search that needs exact keyword matching to find resources. However, users generally have a vague picture of what is stored but forget the exact location and keywords of the resource. According to observations of human associative memory, people tend to remember things from some memory fragments in their brains and these memory fragments are connected by memory cues of user activity context. We developed iMecho (My Memory Echo), an associative memory based desktop search system, which exploits such associations and contexts to enhance traditional desktop search. Desktop resources are connected with semantic links mined from explicit and implicit user activities according to specific access patterns. Using these semantic links, associations among memory fragments can be built or rebuilt in a user's brain during a search. Moreover, our link-based ranking scheme uses these links together with a user's personal preferences to rank results by both relevance and importance to the user. In addition, the system provides a faceted search feature and association graph navigation to help users refine and associate search results generated by full-text keyword search. Our experiments investigating precision and recall quality of iMecho prototype show that the association-based search system is superior to the traditional keyword search in personal search engines since it is closer to the way that human associative memory works.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

Algorithms, Design

Keywords

associative search, desktop search, personalized ranking

CIKM'09, November 2–6, 2009, Hong Kong, China.

1. INTRODUCTION

As most of the digital universe are created by individuals, personal information management (PIM) has become a hot topic. Searching desktop resources, including local files, emails, instant messages, cached web pages, etc., has become an important but time-consuming task [6, 7, 8]. Personal search is different from web search. People generally have a vague picture of what is stored in their computers but they always forget the exact location and keyword of the resource content. Existing desktop search systems such as Google Desktop, Microsoft Windows Desktop Search, and Spotlight for Apple's OS only support keyword search that needs exact keyword matching to find resources.

Considering human memory recall, people appear to remember something through some associative memory fragments left in their brains as well as memory cues relating to the context of information capture or subsequent access. For example, a user wants to find a web page he had read while writing an important report, but he cannot remember the keywords about the web page. Interestingly, he does remember the connection and therefore finds the web page through the report. Psychology research has shown that people often remember things through chains of associations [18]. Our idea is to exploit such associations and contexts to simulate human associative memory to enhance desktop search.

Current desktop file systems do not provide a means to link semantically related files. However, user activities in the desktop, to some extent, reflect his associative memory when he searches resources later. We developed the iMecho (My Memory Echo), an associative memory based desktop search system. Desktop resources are connected in iMecho with semantic links by analyzing user activities and the contexts in which the user works. Using these semantic links, associations among memory fragments can be built or rebuilt in a user's brain during a search. The personalized ranking scheme uses these links together with user's personal preferences to rank results by both relevance and importance. iMecho provides a two-step search paradigm. The user first uses faceted search filters to quickly locate an easy-to-remember intermediate resource from the full-text keyword search results. Then the user can find the target resource by multiple navigation in the association graphs in iMecho.

In some recent PIM systems, e.g. Stuff I've Seen [8], MyLifeBits [10], Haystack [13], Semex [9], Beagle++ [7] and Feldspar [6], contextual cues such as time, author and association information are exploited to search and present personal information. However, the above systems only sup-

port a few kinds of simple predefined associations. iMecho enhances desktop search by building associative links of resources from implicit access patterns of user activity sequences. In iMecho, there are three types of associations: Content-based Associations (CA), Explicit Activity-based Associations (EAA) and Implicit Activity-based Associations (IAA). CAs refer to associations that can be extracted from the content and attributes of resources. EAAs are explicit associations in the sense that they are bound with certain user activities such as jumping to another webpage. IAAs denote some implicit associations between resources, which can be discovered through user access pattern analysis and resource provenance analysis. These semantic associations are designed in accordance with a common mode of thinking in human minds so that the associative memory can be exploited to help find the target resource. Our experiments investigating precision and recall quality of iMecho show encouraging improvements over traditional keyword desktop search.

In summary, iMecho has the following contributions:

- iMecho enhances desktop search by building associative links of resources from user activities. Besides CAs and EAs, iMecho discovers two types of IIAs, *same_task* and *provenance_of*, by mining patterns of user activities. For the *same_task* link, iMecho uses a clustering based algorithm for initial task analysis and an Hidden Markov Model (HMM) method for further optimization.
- Besides the traditional keywords-based searching paradigm, iMecho provides a two-step searching paradigm. The user locates an intermediate resource by keywords and faceted filters in the first step. Then he can use the association graph navigation to find semantically associative resources, which simulates human associative memory.
- iMecho combines a link-based ranking scheme with the common content-based ranking scheme to generate personalized ranking results for every user. Therefore the top results are not only related to the query but also important to the user.

The remainder of the paper is organized as follows: In Section 2 we present the architectural overview of the iMecho system. Section 3 describes the key implementation technologies. Section 4 evaluates its performance. Section 5 discusses the state of the art. Finally, Section 6 concludes this paper.

2. SYSTEM OVERVIEW

2.1 Architecture

There are different kinds of physical resources available on the PC desktop. Currently, iMecho mainly deals with general files and file hierarchies in the file system, email messages and attachments in the email store and offline web pages in the web cache. The architecture of iMecho is given in Figure 1. Like a traditional full-text search engine, the system need crawlers and extractors to parse the full-text content from different data sources. iMecho can extract content from PDF, Word, Excel, Powerpoint, RTF, TXT, Java, CHM as well as ZIP, RAR, and many other archives. The

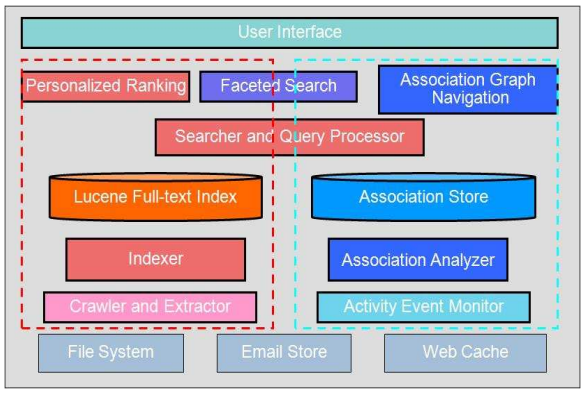


Figure 1: Architecture

indexer module is used to build up indexes for these data sources and to store them in the Lucene full-text index. Files inside ZIP, RAR, CHM and other archives are extracted during indexing and can be preserved for searches. When user inputs keywords in the user interface, the searcher and query processor will find the results through the Lucene full-text index. In addition, a relevance score is computed for each desktop item by ranking module, supporting an enhanced ordering of results based on term frequency.

Besides the traditional desktop search components (left rectangle), iMecho has several additional components (right rectangle) for tracking user activities and generating semantic links. iMecho uses the activity event monitor to record desktop events and association analyzer to create semantic links from the recorded events. Then the resource associations and attributes are represented as RDF triples and stored in an RDF repository. The searcher and query processor module delegates keyword searches and RDF queries to the full-text index and RDF repository respectively. RDF queries are used to retrieve the semantic links of a given resource. iMecho provides two different ways to help users find resources. Faceted search filters are used to refine the full-text search results through multi-dimensional classification. On the other hand, the association graph navigation is used to extend the searching results to associative resources via semantic links. The user interface (showed in Figure 2) visualizes search results and associative context in these two ways as well.

2.2 Activity Event Monitor

In order to construct semantic associations, iMecho needs to monitor user activities on their desktops. Currently, the event monitor can track both system-level and application-level activities. System-level activities include all file system operations and Window related activities such as Window creation, activation and closing. Application-level activities are restricted to specific applications. The monitor provides "plugins" for each of the applications to track user activities on MS-Office documents, Emails (Outlook), Adobe pdf files and cached webpages (Firefox). The plugins also extract application-specific metadata and associations between resources. When an email is received, for example, the Outlook plugin automatically extracts email metadata, e.g. "to", "from" and "subject" from corresponding email fields, and connects the attached documents with the email.

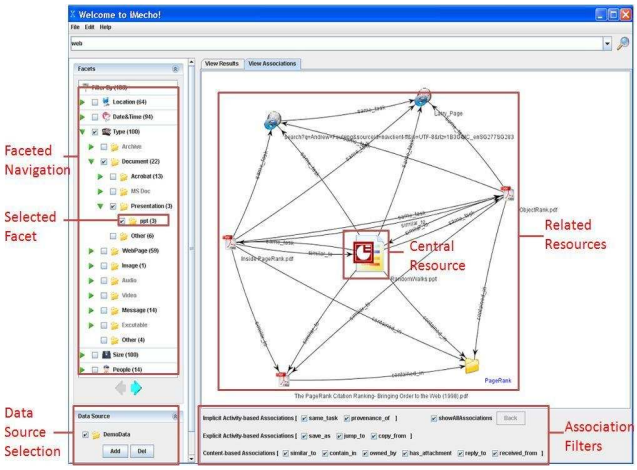


Figure 2: System User Interface

2.3 Association Analyzer

iMecho enhances desktop search by building associative links of resources from user activity context. Through content analysis, user access pattern analysis and lineage analysis, we get the following three types of links.

CA CAs represent the content-based associations created by analyzing the content and attributes of desktop resources. For example, in Figure 3, the *similar_to* link connects two resources that are similar in terms of contents, file names and locations (e.g. same directory). A similarity function regarding these three factors are used to compute the distance between two resources. By analyzing the attributes of email messages and files, we can get *has_attachment*, *reply_to*, *received_from*, *owned_by* and *contained_in* CA links.

EAA EAAs are explicit and deterministic activity based associations. EAA links are set when specific user events are observed. We create three types of EAAs in iMecho: *jump_to*, *copy_from* and *save_as*. Each of them is bound with a type of user activity tracked by the event monitor. For instance when the user goes from one webpage to another, the two pages are connected by a *jump_to* link. When the attachment of an email (or a webpage) is saved as a local file, a *save_as* link will be created connecting the attachment (or webpage) and the file. Similarly the *copy_from* link is created based on "file copy" events.

IAA IAAs denote some implicit and nondeterministic associations between resources, which can be discovered by user access pattern analysis (e.g. the *same_task* link) and resource lineage analysis (e.g. the *provenance_of* link). We find that users tend to access and manipulate different resources to complete a task. We define the *same_task* link to cluster resources by their related tasks. The *same_task* is usually very helpful for the user because we organize our memory in a similar way. The *provenance_of* link can represent the origin of a resource. Users always create multiple versions of a document through several "file copy" and "save as" operations. By analyzing the lineage sequence of *save_as* and *copy_from* activities, we can create the *provenance_of* link between a document and its original copy. Actually, an IAA link is responding to a pattern of user activity sequence.

2.4 Personalized Ranking

Existing desktop search engines only employ the content-based algorithms such as cosine similarity to rank personal desktop resources. However, the content-based ranking could not reflect the user's personal preference. For desktop search, a top ranked document should be not only relevant to the query but also relatively important to the user. The link-based ranking algorithms, such as PageRank, are good ways to find authoritative results. With the established links from user activity, iMecho can combine the two ranking schemes to generate personalized ranking results for every user. The final score of a document is the product of the relevance score and the global authority score. Similar to the ObjectRank [5] algorithm, different weights are manually assigned to different types of links since different links may have different impacts on the global authority score. Therefore more important links, such as *same_task*, are given relatively higher weights. By this means, desktop resources are connected by weighted links. Then we can employ a link-based ranking algorithm to find important results.

Our ranking algorithm is personalized because the user can manually adjust the weights of different types of associations, which explicitly show the user preferences. In addition, user activities can implicitly influence the ranking computation since some semantic links are created by analyzing user access patterns and some parameters in the algorithm are decided by the user access frequency as well.

2.5 Faceted Search

Faceted Search enables users to navigate in a multi-dimensional information space by combining text search with a progressive narrowing of choices in each dimension. In iMecho, we apply faceted search to desktop search to refine results from keyword search. It combines the effectiveness and flexibility of full-text search with the ease-of-use of faceted navigation: the ability to find information based on attributes such as its location, file type, modification dates, file size, author, etc. Unlike web facets, desktop facets in iMecho are well organized into a facet tree, which further classifies facets of the same taxonomy. In addition, users can select multiple facet terms from different taxonomies at the same time to quickly filter the result set. In iMecho, there are two kinds of desktop facets, predefined desktop facets (e.g. file size, modified date, file type) and dynamically generated facets, such as the senders of email messages, in the sense that their values vary according to the result set.

2.6 Association Graph Navigation

In desktop search, users tend to associate resources in their minds and try to follow these associations when looking for specific resources. There are some examples here. "I remember that I have browsed several webpages weeks ago, but I forgot either its URL or its content. On the other hand, I remember that when I browsed the page, I was writing a technical report, and I remember some keywords of that document", "Several images were downloaded from attachments of an email several months ago. Now I want to find a paper with the same folder as these images. But I could not remember either its name or its directory. I only know that Peter sent the email and talked about desktop search topic". iMecho manages to process such searches. The association graph shows all directly related resources to a specific resource (showed in Figure 2). A user can first locate an intermediate resource by keywords and faceted search,

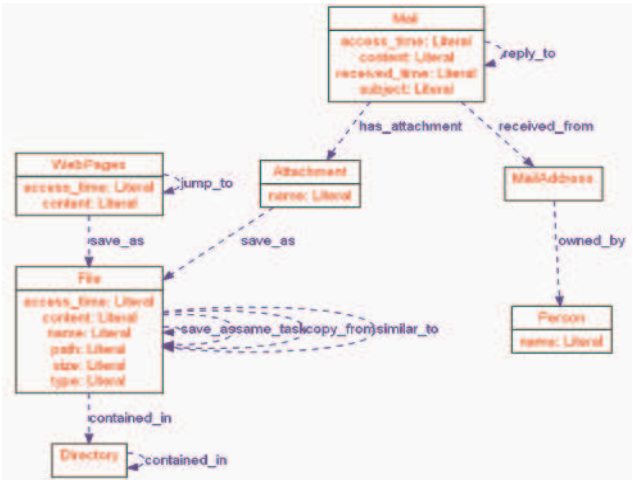


Figure 3: The Desktop Ontology

and then switch to the association graph to get its related resource. When the user selects one of the associative resources, say resource r, the graph will be updated and all related resources of r will be displayed. By this means, the user can navigate in the association graph, which simulates the associative memory in the mind of the user.

3. KEY IMPLEMENTATION TECHNOLOGIES

3.1 Desktop links analysis

Desktop search could potentially profit from a lot of implicit and explicit semantic information available in emails, folder hierarchies, browser cache contexts and others (contextual metadata, provenience of information as well as sophisticated classification hierarchies). Figure 3 shows the desktop ontology in iMecho, which represents desktop resource attributes and semantic associations among emails, folder and file hierarchies, cached web pages, etc.

As mentioned above, associations among desktop resources include CAs, EAs and IAs. Since iMecho is a desktop search engine rather than a data integration prototype, we mainly focus on associations related to user activities (i.e. EAAs and IAAs), in particular, the IAA links. IAAs can be mined by recognizing specific patterns in the user log. In this section, we will describe the task mining algorithm and its optimization in generating the *same_task* IAA link in detail. To the best of our knowledge, this work remains unexplored up until now.

Users tend to access and manipulate different resources to complete a task. For instance, when writing a paper, a user may look for related presentations, references, web pages and emails from colleagues. These resources are all somehow related to this task. This type of links is defined as *same_task*, which corresponds to a special pattern of user activity sequences. A task can be taken as a series of user activities with a specific goal. Two document resources are associated via *same_task* if they are involved in a same working task. To generate the *same_task* links, we propose a clustering based algorithm for initial task mining and a Hidden Markov Model based method for further optimization.

A Clustering based Algorithm for Initial Task Mining

A user always switches between different tasks in a given period of time. For instance, a researcher starts to read a paper in the Monday morning. Then he gets a email regarding an upcoming presentation. After finishing the slides he gets some good ideas and turns to write a working draft. In the end he goes to the paper he read in the first place. This example shows how difficult it is to decide which resources are related to the same task.

We try to ease the problem by introducing the notion of "*key resources*". A *key resource* is the goal of a task. All other resources related to this task work as references or appendixes to the *key resource*. It can be a report, a read-only paper, an Email message and so on.

Definition 1. A task T is defined as a set of resources. If resource a is accessed by the user when he is accomplishing task T, we denote $a \in T$.

Resource a may belong to several tasks, which means a is used several times for different purposes.

Definition 2. The *lifecycle* of a resource a (denoted as $L^k(a)$, k is the occurrence index of a in the log) is the gap between opening time ($O^k(a)$) and closing time ($C^k(a)$) of its time window, i.e., $L^k(a) = C^k(a) - O^k(a)$. $|L^k(a)|$ is the length of $L^k(a)$. The index k can be omitted if no misunderstanding occurs.

If $L(b)$ is completely covered by $L(a)$, then $L(b) \subset L(a)$ or $L(a) \supset L(b)$. If $L(a)$ and $L(b)$ are overlapped, which means that the user opens b before closing a or vice versus, $L(a) \sim L(b)$. If $L(a) \supset L(b)$ or $L(b) \supset L(a)$, then $L(a) \sim L(b)$.

A resource may have several *lifecycles* in a given period of time with different k.

We make several assumptions about key resources.

Assumption 1. There is only one key resource in a given task. The key resource of task T is denoted as $K(T)$.

According to the definition, the key resource serves as the goal of the task. We assume that a user has one task at one time. Therefore there is only one key resource in a task.

Assumption 2. A key resource has the longest lifecycle among all the resources in the corresponding task, i.e., $\forall a \in T, |L(K(T))| \geq |L(a)|$.

In most cases the user opens the key resource and keep editing it until it is finished.

Assumption 3. Task T starts when its key resource $K(T)$ is opened and T stops when $K(T)$ is closed. All the overlapping resources of $K(T)$ go to task T, i.e., $\forall a$, if $K(a) \sim K(T)$ then $a \in T$.

This assumption comes from the observation that the user usually opens resources that are related to the key resource he is working on. When the key resource is closed, the current task is accomplished. Of course the user may open the key resource in the future but that is treated as another task so far.

Because usually when a user is working on a task, he opens

all resources involved. Therefore what we should do is to decide the boundary of each task and get the corresponding resources inside the boundary. A initial algorithm is used to discover tasks and create the *same_task* links. We cluster resources by their lifecycles. If the lifecycle of resource a is completely covered by that of resource b, then a and b go to one cluster. The resource that has the longest lifecycle is the *key* of the cluster. If the lifecycles of two keys are overlapped, we use some rules to decide whether the two clusters should be merged or not. When no clusters can be merged, each cluster is output as one task. All resources in the cluster are connected by the *same_task* link. Since the boundary of a task is determined by its key resource according to assumption 3, we should get all the key resources first. To make sure that these key resources satisfies assumption 2, they should be ranked by their lifecycles in the first place. The initial task mining algorithm is showed in Algorithm 1.

Algorithm 1. TaskMining
Input: Resource list $\mathcal{R}$, Lifecycle map $\mathcal{L}$. Suppose $r \in \mathcal{R}$, $\mathcal{L}(r)$ stands for the lifecyle of r.
Output: Task list $\mathcal{T}$

```
T = φ
WHILE (TRUE)
    IF |L| = 0
        OUTPUT T
        EXIT
    END IF
    m = −1;
    FOR EACH r ∈ R
        IF (m < |L(r)|)
            m = |L(r)|, R_m = r
        END IF
    END FOR
    L.remove(R_m)
    t = new Task
    FOR EACH r ∈ R
        IF (L(r) ~ R_m)
            t.insert(r)
        END IF
    END FOR
    T.insert(t).
END WHILE
```

Sometimes the lifecycles of resources are too short that few of them are overlapped. It usually happens when the algorithm is coming to an end because lifecycles are sorted in descending order. A simple solution is to set a threshold for the minimal size of a task. The default value of this threshold is 2, which means all tasks with less than two resources will be removed from the output.

An HMM-based Optimization for Task Mining After getting the initial tasks results, we optimize the tasks using a well-studied mathematical model, Hidden Markov Model (HMM), which is proved to yield task mining results that are closer in line with the user logs.

HMM is a widely-used stochastic model in which the set and sequence of states in a system can be estimated by studying the observed behaviors of the system. In desktop systems, states can be represented as tasks and the observed behaviors are user activity logs on desktop resources. Users tend to access different resources to complete a task and usually keep switching between different tasks.

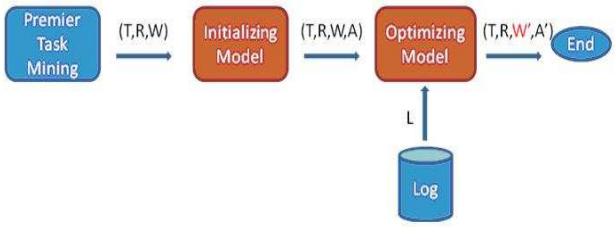


Figure 4: The Optimization Process of Task Mining

The output of the initial task mining can be represented as a triple (T, R, W). T stands for the set of mined tasks. R is the set of resources related to T. W is the weight matrix. W_{ij} is the weight that resource R_j is in task T_i. The weight W_{ij} is calculated as following:

$$W_{ij} = \begin{cases} \frac{1}{N_T(i)} & : \quad R_j \in T_i \\ 0 & : \quad otherwise \end{cases}$$

Here $N_T(i)$ is the number of resources in task T_i. If $W_{ij} > 0$, resource R_j is related to task T_i. The HMM model can be represented as $\lambda = (T, R, W, A)$, in which T, R and W are the inputs of the initial task mining step. A is the transfer matrix. A_{ij} denotes for the possibility that the user switches from task T_i to task T_j. The user log L is a sequence of the elements of R. $|L|$ stands for the length of the log.

The optimized task mining process is shown in Figure 4. The initial task mining results are generated in the first place. Then an HMM is built on the results. After optimizing the model with user log L, the weight matrix W and transfer matrix A are updated to W' and A' respectively. In the end the optimized HMM model $\lambda' = (T, R, W', A')$ is output. The task set T and the resource set T is not changed during the optimization process. But the weight matrix W is updated as W' , which is the key of the task mining result. The last step of task mining is to cluster all resources according to W'. For task T_i, resource $R_j \in T_i$ if $W'_{ij} \geq N_T(i)$. $N_T(i)$ is the number of resources of T_i before optimizing.

The goal of the optimization is to make the model more precise. The Baum-Welch algorithm [8] is used to optimize the parameters of HMM to make it "closer" to the log. It is proved that $P(L|\lambda') \geq P(L|\lambda)$. In other words, the optimized model is more likely to generate the observed user log than the initial model. In addition, the transfer matrix A is less important in the optimization process. Uniform initial estimate of the matrix A is adequate for giving useful optimization of these parameters in almost all cases.

3.2 Personalized Ranking

The ranking algorithm is one of the key technologies to search engines. Current desktop search products, such as Google Desktop and Windows Desktop Search, are now comparable to first generation web search engines, which provided full-text indexing, but only relied on textual information retrieval to rank their results, e.g. content-based ranking algorithms. These algorithms use heuristic distance functions to compare how "close" the query and indexed documents are. The most frequently-used distance function is called *cosine similarity*, which uses the cosine of the angle between the two term vectors as the distance. Content-based

ranking algorithms, no matter which distance function they employ, only rank documents by their "relevance" to the given query. It is widely accepted that users are also interested in the "importance" of the pages. The more important the page is, the higher rank in the result set it should have. PageRank [17] is one of the most famous algorithm which ranks pages by their importance. It calculates the probability that each node is visited by the "Random Surfer" model, which is used as the score of importance. The "Random Surfer" model can simulate the behavior of a web user such that the ranking score of a page is approximately the probability that it is visited by the user in his tour on the Web. The "Random Surfer" model assumes that the surfer follows three rules when navigating on the Web.

1. He starts with a random page.
2. He may click one of the links in the existing page with equal probabilities Or
3. He is not interested in the current page and its links so he goes to a random page.

However, current desktop search engines can not benefit from this ranking scheme because there are no established links among those isolated desktop resources (files, emails, etc.). iMecho overcomes this problem by building semantically associative links among desktop resources from the analysis of user activity context. Then a new personalized link-based ranking algorithm - iRank is formed based on such link structures together with the user preferences. Different from the the random surfer model in PageRank, iRank employs the following user model:

1. He starts with a random resource.
2. He may either access one of the resources that are related to the current one with certain probabilities. The probabilities are in direct proportion to the importance of the outgoing links.
3. Or he goes to a resource with a probability that is proportional to the frequency the resource appears in the user log.

In iMecho, a top ranked document should be not only relevant to the query but also relatively important to the user in the result set. Specifically, iRank uses a combination of the TF*IDF score returned by Lucene and an extension of PageRank. Given query q, the ranking score of a resource e is denoted as $S^q(e)$. Suppose $E^q(e)$ is the relevance score between e and q, $R(e)$ is the importance score of e, then

$$S^q(e) = E^q(e) * R(e)$$

$E^q(e)$ is given by Lucene at the query time. And $R(e)$ is calculated by the link-based algorithm, which will be introduced later. As PageRank, iRank is an offline algorithm which runs periodically on the dataset.

For ranking personal desktop information, iRank has two advantages over PageRank.

- Users can manually assign different weights to different types of links such that important links contribute more to the final results. The weights explicitly show the user preferences.
- Our algorithm makes full use of the implicit user feedbacks by tuning ranking parameters according to user activity logs. So different users will get different ranking results given the same query.

Weighted Links There are different types of links in iMecho. iMecho extends the desktop ontology of Figure 3 by adding weights and edges in order to express how importance propagates among the entities and resources inside the desktop ontology. For example, authority of an email is split among the sender of the email, its attachment, and the email to which it was replied. So, if an email is important, the sender might be an important person, the attachment be an important one. And the previous email in the thread hierarchy also become important. In iMecho, different weights are manually assigned to different types of links. Every edge from the ontology graph is split into two edges, one for each direction. This is motivated by the observation that authority potentially flows in both directions and not only in the direction that appears in the original desktop ontology (if we know that a particular person is important, we also want to have all emails we receive from this person ranked higher).

Figure 5 shows the links between different types of resources in Imecho and weights of these links as well. The user is more likely to follow the links with higher weight according to our model. Therefore nodes pointed by these links are more likely to be visited by the user.

There are a couple of issues should be noted in Figure 5.

- The number of links in Figure 5 is twice as many as that in Figure 3. For any link in Figure 3(dashed line), there is an opposite *rev_* link (real line) in Figure 5. Let's take *copy_from* as an example. Suppose file A is a copy of file B, i.e. $A \overset{copy\_from}{\longrightarrow} B$, then the importance of A may affect B and vice versus.
- The opposite link may have different weight comparing to the original link. For instance, the weight of *copy_from* is 0.7 and that of *rev_copy_from* is 0.2. Suppose file A is a copy of file B, A may have some new contents which are not in B. If the user has opened B, he is probably interested in A too. However, if the user has read A, he may not be so interested in B because A is newer.
- The weights of all outgoing links of a node in Figure 5 are not summarized to 1. Figure 5 is the global view of the "importance" of all types of links so that different kinds of links can be comparable in terms of importance. Users can easily find that *same_task* is more important than *owned_by*. If the weights are normalized, the importance of links from different resources (such as *same_task* and *owned_by*) can not be compared.

Personalization Different authority transfer weights express different preferences of the user, translating into personalized ranking. As mentioned before, the user activities that influence the ranking computation have also to be taken into account, which translates to assign different weights to different contexts. The weights explicitly show the user preferences. In addition, the ranking is implicitly personalized by analyzing the user activity logs. Before running iRank, we study the activity logs and calculate the access frequency of each resource. According to our user model, the

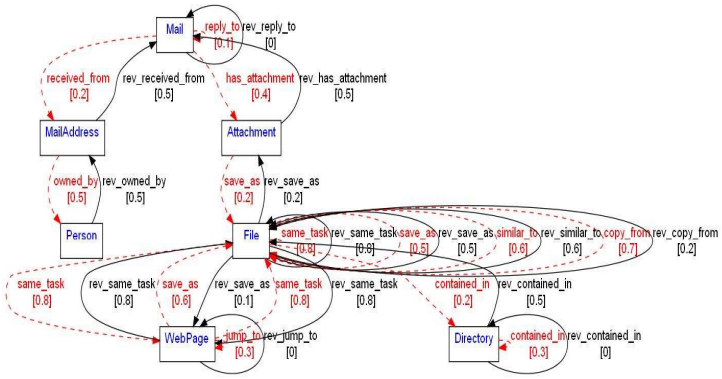


Figure 5: Weighted Desktop Ontology

user prefers to the most frequently visited resource when he decides not to follow the existing links. Then we use this access frequency information to adjust our ranking algorithm so these user-preferred resources will get higher ranks.

Computation Algorithms The computation of rankings in iRank is based on the link structure of the resources as specified by the defined ontologies and the corresponding metadata. G is the directed graph of connected desktop resources. Suppose n is the number of nodes in the graph G, $\vec{r}_k(n+1)$ denotes the access probability vector, i.e. the importance score vector, after k rounds, there is

$$\vec{r}_0 = (1/n, \ldots, 1/n)^T \quad (1)$$

$$\vec{r}_k = (d * \vec{A^T} + (1-d) * E) \times \vec{r}_{k-1} \quad (2)$$

Here d is the dumping factor and it is set to 0.85 in general cases. Matrix A also known as the *Transition Matrix*, which denotes the weighted graph of indexed files. After some algebra, equation 2 turns to:

$$\vec{r}_k = d * \vec{A^T} \times \vec{r}_{k-1} + (1-d) * \vec{E} \quad (3)$$

$\vec{E}$ is the unit matrix, a *Personalization Vector* and $e_i = c_i / \sum_i c_i$ (c_i is the number of occurrences of resource i in the user log). The random jump to an arbitrary resource in the resource graph is modeled by $\vec{E}$ in the Random Surfer model, which shows the probabilities that pages are visited by the surfer when he is not following the links of the current page. This usually happens when the user gets bored with the page he is reading and then goes to a familiar site. In web search, users go to different pages with equal probabilities so $\vec{E}$ is a uniform vector in PageRank. The result is that every user will get exactly the same ranking list given the same query. While for desktop search, with the help of user logs, we can easily get the preference of the user and set the value of $\vec{E}$ according to the user access frequency. In iRank, by appropriately modifying the $\vec{E}$ with different weights assigned to different data sources according to their access frequency, it can model user's preference and personalization.

The iRank algorithm consists of two parts. The first is the initialization phase, which initializes the transition matrix A. The second is the iteration phase, which iteratively calculates ranking scores of every node.

- **The Initialization Phase** Suppose $i \xrightarrow{k} j$ means directed edge $i \to j$ is of type k , d_i^k means the number of outgoing type k links from node i and w^k stands for the weight of type k links in Figure 5, then transition matrix $\vec{A}$ is computed by Algorithm 2.

Algorithm 2: Transition Matrix Initialization

Input: Resource Graph G and Weighted Ontology Graph W (w^k and d_i^k)

Output: Transition Matrix A

FOR i FROM 1 TO n

  FOR j FROM 1 TO n

    IF $i \to j$ is not an edge of G

      $A_{ij} = 0$ // i and j are not connected

    ELSE

      $i \xrightarrow{k} j$

      $A_{ij} = w^k / d_i^k$ // Different from PageRank

    END IF

  END FOR

  $W_i = \sum_{j=1}^{n} A_{ij}$

  FOR j FROM 1 TO n

    $A_{ij} = A_{ij} / W_i$ // Normalize Each Vector

  END FOR

END FOR

A is the adjacency matrix which connects all available instances of the existing context ontology on the user's desktop. The weights of the links between the instances correspond to the weights specified in the weighted desktop ontology divided by the number of the links of the same type. When instantiating the ontology for the resources existing on the user's desktop, the corresponding matrix A will have element values which can be either 0, if there is no edge between the corresponding entities in the graph, or the weight assigned to the edge determined by these entities, in the ontology, divided by the number of outgoing links of the same type. To help understand the algorithm, we give a simple example. Suppose node i has 5 outgoing links, 2 of them are of type A and others are of type B. The weights of type A and B are 0.4 and 0.6 respectively. Then after normalization the weight of every outgoing link is 0.2.

- **The Iteration Phase** After the initialization of $\vec{A}$ we go to the iteration phrase showed in Algorithm 3.

Algorithm 3 iRank Computation

Input: Transition Matrix A, dumping factor d, Personalization Vector e and the threshold ϵ.

Output: Ranking score vector $\vec{r}$

$\vec{r} = (1/n, 1/n, \ldots, 1/n)$

$\vec{r'} = (0, 0, \ldots, 1)$

WHILE TRUE

  $\vec{r'} = d * \vec{A^T} \times \vec{r} + (1-d) * \vec{e}$

IF ($|\vec{r'} - \vec{r}| \leq \epsilon$) // has converged already
BREAK
ELSE $\vec{r} = \vec{r'}$
END IF
END WHILE

In the iRank algorithms, we note that the initial value of $\vec{r}$, i.e. $\vec{r}_0$ in Equation 1, does not affect the final ranking score. But it does affect the rate of convergence. In addition, the personalization vector MUST NOT have any zero elements, otherwise iRank might NOT converge. Therefore, if resource i is not accessed according to the user log, i.e. $c_i = 0$, we set $e_i = 1/\sum_i c_i$.

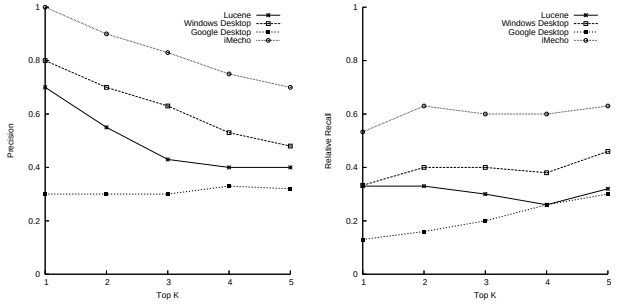
4. PERFORMANCE EVALUATION

4.1 Experimental Setup

We did an initial evaluation of our system by conducting a small scale user study. Five of students in Fudan University provided a set of their locally indexed documents, some of which they received as attachments to emails. The average data set contains 9431 desktop files in 1019 directories. The average directory depth is 9 with the longest being 15. On average, directories contain 10.3 sub-directories and files, with the largest containing 241 ones. 75% of the files are smaller than 16KB, and 95% of the files are smaller than 40 KB. The largest file is of size 21.5MB. The user log produced by the event monitor records totally 1601 desktop events. Then, each user issues search queries, related to their activities, over the indexed dataset. In total, 10 queries were issued ("beagle indexing", "beagle ranking", "PageRank model", "random walk", "desktop search application" are some sample queries). The average query length was 1.9 keywords, which is slightly more than the average of 1.7 keywords reported in other larger scale studies (e.g. in [8]). For comparison purposes, we sent each of these queries to Windows Desktop, Google Desktop and the Lucene-based desktop search systems (content-based ranking), and the iMecho system (containing semantic links from user activity context and personalized ranking) respectively. For every query, each user rated the top 5 output results for each system using grades from 0 to 1 where 0 for an irrelevant result, 0.5 for a relevant one, and 1 for a highly relevant one.

4.2 Evaluation of Personalized Ranking

Even when semantic links are integrated as part of a search system, the traditional measures from information retrieval theory can and should still be applied when evaluating system performance. We therefore used the ratings of our users to compute average precision and relative recall values at each output rank. Both measures can be calculated at any rank k, i.e., considering only the top k results output by the application. Restricting the calculation of precision and recall to various ranks is useful in order to investigate the quality of the system at different levels. It is a common practise to focus only on the top-k results. In the experiment we follow the practise to use the top-5 results only. Another important aspect is when calculating the recall measure, we need to know the total number of relevant results, which is extremely difficult to know. We use the total number of unique relevant results returned by the four ranking schemes. For every query, each returns 5 results. So the overall relevant number is 20 at most. That is the base of relative recall [7].



(a) Average precision (b) Average relative recall

Figure 6: Comparisons of average precision and relative recall

We averaged the precision values at each rank from one to five for all 10 queries submitted by our users. The results are depicted in Figure 6(a). We first notice that the Desktop Search with Lucene, Google and Windows Desktop are poor, containing more qualitative results towards rank 4 to 5, rather than at the top of the result list. This is, in fact, explainable since Lucene, Google and Windows Desktop only uses TF*IDF to rank its results, thus missing any kind of global importance measure for the desktop resources. On the contrary, our iMecho system enhanced with semantic metadata and links, performs much better. An important reason for this high improvement is that metadata are mostly generated for those resources with high importance to the user, whereas the other automatically installed files (e.g., help files) are not associated with metadata, and thus ranked lower. When we have metadata describing a desktop item, more text is inherently available to search for, and thus this item is also easier to find. In addition, personalized ranking pushes our resources of interest more towards the top of the list, yielding higher desktop search output quality. We drew similar conclusions with respect to the average recall values (depicted in Figure 6(b)). The recalls of the Lucene and Google desktop search system are very low, whereas that of iMecho system is almost twice better (owing to the additional semantic links mined from user activity context). We thus conclude that iMecho significantly increases its recall (as semantic links and other metadata usually represent additional, highly relevant text associated to each desktop file), whereas adding desktop ranking further contributes with a visible improvement in terms of precision.

4.3 An Example of Task Mining

iMecho employs a task mining algorithm to generate the *same_task* associations between resources. To give the reader a feeling of the tasks detected by the algorithm, Table 1 shows the tasks mined from one of the users' (user A) activity log. To save space, only relative path of the resource is used. The descriptions of the tasks are added after an interview with user A about the semantics of these tasks,

Table 1: Tasks mined from user activity log for generating the *same_task* links.

Task	Resources involved	Brief description
T1	Documents/Beagle/RDFRepository/RDFRepository_JavaDoc.html Tools/openrdf-sesame-2.1.3/docs/users/ch07.html Tools/openrdf-sesame-2.1.3/docs/users/ch08.html Materials/Lucene_in_ Action.pdf Documents/Beagle/Minack 2008_LuceneSail.pdf Materials/RDF/RDF_Query_Language_Compare.htm Tools/openrdf-sesame-2.1.3/docs/users/index.html Tools/openrdf-sesame-2.1.3/docs/users/ch09.html Tools/openrdf-sesame-2.1.3/docs/users/ch06.html	*Lucene* and *Sesame* studying
T2	Papers/PageRank/RandomWalks.ppt Papers/PageRank/The PageRank Citation Ranking- Bringing Order to the Web (1998).pdf Downloaded/auth.pdf Papers/PageRank/Inside PageRank.pdf Papers/PageRank/ObjectRank.pdf	Survey of popular ranking schemes
T3	Documents/Beagle/Beagle_Doc/Backend_Tutorial.htm Materials/BeagleInvestigation/BeagleInv.pdf Documents/Beagle/Beagle_Doc/Indexing_Data.htm Documents/Beagle/Beagle_Doc/Architecture_Overview.htm Documents/Beagle/WhitePapers/Beagle++ - Indexing and Querying your Desktop.pdf Documents/Beagle/Beagle_Doc/Filter_Tutorial.htm	Investigation of *Beagle*'s indexing module
T4	Papers/SemanticDesktop/chi08-feldspar.pdf Slides/Comparions of Desktop Search Tools.ppt Papers/Beagle++/ESWC06- Beagle++.pdf Slides/semantic search.ppt Slides/Beagle++Toolbox.ppt Papers/Beagle++/ESWC05-Activity-Based Metadata for Semantic Desktop Search.pdf	Investigation of several desktop search tools
T5	Documents/Beagle/WhitePapers/Beagle++Ranking.pdf Documents/Beagle/Beagle_Doc/Searching_Data.htm Documents/Beagle/Beagle_Doc/Backend_Tutorial.htm Materials/BeagleInvestigation/BeagleInv.pdf Documents/Beagle/Beagle_Doc/Architecture_Overview.htm Papers/PageRank/ObjectRank.pdf	Investigation of *Beagle*'s ranking module

who was recently engaged in a project on desktop search engines. Specifically, in $T1$, *Lucene* is a full-text search engine, and *Sesame* is a well known RDF repository incorporating efficient RDF storage and query processing functionalities. *LuceneSail* is simply a combination of these two technologies. These resources are all related to that task. Other discovered tasks also make sense. $T2$ is the writing of a survey of popular ranking schemes widely used by current web search engines; $T3$ and $T5$ are the writing of two reports about *Beagle*, which is a desktop search engine on the Linux platform. They focus on Beagle's indexing and ranking methods respectively. Finally, $T4$ is an investigation of different desktop search tools. Tasks mined from other user's logs also give positive results and we omit the details here due to space limitation.

5. RELATED WORK

The difficulty of accessing information on our computers has prompted several first releases of desktop search applications recently. The most prominent examples include Google desktop search [3] and Microsoft Windows Desktop Search [4] (proprietary, for Windows) and the Beagle open source project for Linux [2]. Yet they include no metadata and associations whatsoever in their system, but just a regular text-based index. Apple Inc. integrated an advanced desktop search application (named Spotlight Search [1]) into their operating system, Mac OS Tiger. Even though they also added semantics into their tool, only explicit information is used, such as file size, creator, or metadata embedded into specific files. While this is indeed an improvement over regular search, it still misses contextual information often resulting or inferable from explicit and implicit user actions.

Some PIM systems have been constructed in order to facilitate re-finding of various stored resources on the desktop. Stuff I've Seen [8] for example provides a unified index of the data that a person has seen on her computer, regardless of its type. Based on the fact that the user has already seen the information, contextual cues such as time, author, thumbnails and previews can be used to search for and present information. Similarly, MyLifeBits [10] targets storing locally all digital media of each person, including documents, images, sounds and videos. They organize these data into collections and, like us, connect related resources with links. Haystack [13] emphasizes the relationship between a particular individual and her corpus. It is similar to our approach in the sense that it automatically creates connections between documents with similar content and it exploits usage analysis to extend the desktop search results set. Feldspar [6] is one kind of link-based desktop search prototype with new interface. Users can propose associative queries via a well-designed interface rather than simple keywords. Because user activities are not tracked in Feldspar, only CAs are sup-

ported. Semex [9] is also link-based person information system. It employs a fancy reference reconciliation algorithm to integrate data from different sources and construct content-based associations by extracting metadata. As Feldspar, it only supports CAs and has not activity-based associations mining from user access patterns. Beagle++ [7] is another kind of link-based desktop search prototype, which is closer to our work. It proposed various activity specific heuristics to generate links between resources that associates desktop resources (i.e. local files, emails and cached web pages) via CAs and EAAs. Beagle++ also logs user activities such as attachment saving, file downloading to generate EAAs, but it only focuses on associations from predefined user actions between web pages, email messages and files, not mining from a sequence of user access activities. Their approach was limited to specific desktop contexts (e.g., publications, or web pages), whereas in our methods we explore much more general sources of linkage information such as file access patterns, which are applicable to any desktop resource.

All of the above systems only support a few kinds of simple predefined associations, not semantic associations mining from special patterns of user activity sequences. In iMecho, by exploiting file access patterns and user implicit tasks, many more associations are constructed to better simulate human associative memory in the search. In addition, iMecho combines association navigation with faceted search to incrementally filter search results.

Personalization techniques have been developed in diversified ways for web search. In a nutshell, the techniques can be classified into three categories, namely, content based personalization, link-based personalization, and function-based personalization [14]. Content-based personalization deals with the "relevance" measure of Web pages and the user's queries. In order to manage user interests, a content-based personalization technique is used to construct user profiles, which store user interests derived from each user's search history [15]. Link-based personalization performs personalization based on link analysis techniques. They redefine the importance of Web pages according to different users' preferences such as bookmarks as a set of preferred pages [11]. The function-based personalization first discovers user preferences on the search results from clickthrough data and then the ranking function is optimized according to the discovered preferences [12, 16]. However, these personalized techniques still miss contextual information often resulting or inferable from explicit and implicit user activities. For personalization in desktop search, the desktop environment is comparably "limited" in the sense that we will be able to describe most relevant contexts more easily. It is possible to obtain the complete trace of user activity on his desktop, and therefore his accurate interests, goals, and preferences can be discovered during a search.

6. CONCLUSIONS

We developed an associative memory based desktop search system, iMecho, which enhances the conventional full-text keyword search with semantic associations discovered from user activity contexts. In addition, the system provides the faceted search and association graph navigation to help users refine and associate search results generated by the keyword search. iMecho is superior to the traditional keyword based search engines because it is closer to the way that human associative memory works.

7. REFERENCES

[1] Apple spotlight search. http://developer.apple.com/macosx/tiger/spotlight.html.

[2] Gnome beagle desktop search. http://www.gnome.org/projects/beagle/

[3] Google desktop search. http://desktop.google.com/

[4] Windows desktop Search. http://www.microsoft.com/windows/products/winfamily/desktopsearch/default.mspx

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

[6] D. H. Chau, B. Myers and Faulring, A. What to Do When Search Fails: Finding Information by Association. In CHI 2008.

[7] P. A. Chirita, S. Ghita, W. Nejdl, and R. Paiu. Beagle++: Semantically enhanced searching and ranking on the desktop. In ESWC 2006.

[8] 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.

[9] X. Dong and A. Halevy. A Platform for Personal Information Management and Integration. In CIDR 2005.

[10] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. Mylifebits: fulfilling the memex vision. In ACM Conference on Multimedia 2002.

[11] T. Haveliwala. Topic-sensitive pagerank. In WWW 2002.

[12] T. Joachims. Optimizing search engines using clickthrough data. In ACM SIGKDD 2002.

[13] D. R. Karger. Haystack: A Customizable General-Purpose Information Management Tool for End Users of Semistructured Data. In CIDR 2005.

[14] Y. Ke, L. Deng, W. Ng, D. L. Lee. Web dynamics and their ramifications for the development of Web search engines. Comput. Netw. J. (Special Issue on Web Dynamics), 2005 (50).

[15] F. LIU, C. YU, and W. MENG. Personalized web search for improving retrieval effectiveness. IEEE Trans. Knowl. Data Eng., 2004 (16).

[16] W. NG, L. DENG, and D. L. LEE. Mining user preference using spy voting for search engine personalization. ACM Transactions on Internet Technology, 2007, 7(4).

[17] Larry Page, et al. The PageRank Citation Ranking: Bringing Order to the Web, Tech Report, 1998.

[18] E. Tulving and D. Thomson. Encoding specificity and retrieval processes in episodic memory. Psychological Review 80, 1973.

[19] L. Welch. Hidden Markov Models and the Baum-Welch Algorithm, In IEEE Information Theory Society Newsletter 2003, 53(4).