Fast, Flexible Filtering with *Phlat* — Personal Search and Organization Made Easy

Edward Cutrell, Daniel C. Robbins, Susan T. Dumais, Raman Sarin
Microsoft Research
1 Microsoft Way, Redmond, WA 98052, USA
{cutrell, dcr, sdumais, ramans} @microsoft.com

ABSTRACT

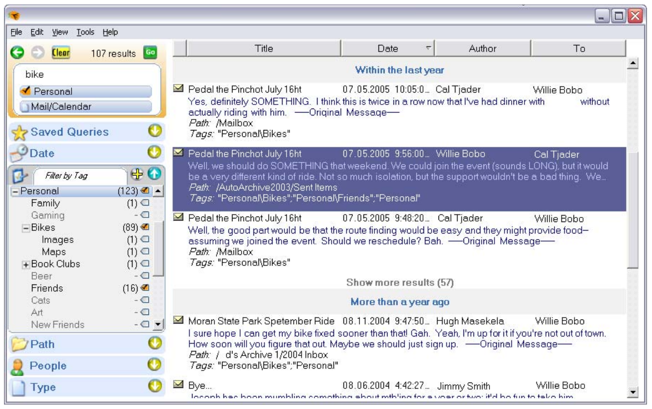
Systems for fast search of personal information are rapidly becoming ubiquitous. Such systems promise to dramatically improve personal information management, yet most are modeled on Web search in which users know very little about the content that they are searching. We describe the design and deployment of a system called *Phlat* that optimizes search for personal information with an intuitive interface that merges search and browsing through a variety of associative and contextual cues. In addition, Phlat supports a unified tagging (labeling) scheme for organizing personal content across storage systems (files, email, etc.). The system has been deployed to hundreds of employees within our organization. We report on both quantitative and qualitative aspects of system use. Phlat is available as a free download at http://research.microsoft.com/adapt/phlat.

Author Keywords
Personal information management, user interfaces, interactive information retrieval, tagging.

ACM Classification Keywords
H5.2. Information interfaces and presentation (e.g., HCI).

INTRODUCTION

Search engines are a popular and ubiquitous tool for finding information, especially on the Web. With a few keystrokes, desired information appears as if by magic from billions of Web pages. Now, search tools are beginning to enjoy widespread use for finding personal information. Indeed, a number of systems for personal search are available on today's PCs. Systems from Microsoft (desktop.msn.com), Yahoo! (desktop.yahoo.com, formerly X1), Google (desktop.google.com), Copernic (copernic.com) and others are all available as free downloads, and this search



Title	Date	Author	To
Within the last year			
Pedal the Pinchot July 16ht	07.05.2005 10:05:0...	Cal Tjader	Willie Bobo
Within the last year			
Pedal the Pinchot July 16ht	07.05.2005 9:56:00...	Willie Bobo	Cal Tjader
Pedal the Pinchot July 16ht	07.05.2005 9:48:20...	Cal Tjader	Willie Bobo
More than a year ago			
Moran State Park Spetember Ride	08.11.2004 9:47:50...	Hugh Masekela	Willie Bobo
Bye...	08.06.2004 4:42:27...	Jimmy Smith	Willie Bobo

Yes, definitely SOMETHING. I think this is twice in a row now that I've had dinner with without actually riding with him. ——Original Message——
Path: /Mailbox
Tags: "Personal\Bikes"

Well, we should do SOMETHING that weekend. We could join the event (sounds LONG), but it would be a very different kind of ride. Not so much isolation, but the support wouldn't be a bad thing. We...
Path: /AutoArchive2003/Sent Items
Tags: "Personal\Bikes";"Personal\Friends";"Personal"

Well, the good part would be that the route finding would be easy and they might provide food– assuming we joined the event. Should we reschedule? Bah. ——Original Message——
Path: /Mailbox
Tags: "Personal\Bikes"

I sure hope I can get my bike fixed sooner than that! Gah. Yeah, I'm up for it if you're not out of town. How soon will you figure that out. Maybe we should just sign up. ——Original Message——
Path: / d's Archive 1/2004 Inbox
Tags: "Personal\Bikes";"Personal"

Figure 1. The Phlat interface.

functionality is being built into the newest generation of operating systems for PCs (e.g., Apple's Spotlight for Tiger OSX and Microsoft's Vista OS). However, searching for personal information is different from searching on the Web in a number of ways. Perhaps the biggest difference is that personal information is, well, *personal*. People are familiar with a host of details and characteristics about their information, as well as the contexts surrounding their use of it. As a result, searching for personal information can be far richer than a simple keyword search of content. Search for personal information can leverage all kinds of details associated with the content. The challenge lies in creating a user interface that exploits the wide and varied details and connections between information sources that users may remember, while maintaining the simplicity of keyword search that makes Web search so powerful and easy.

We created ***Phlat*** (see Figure 1) to explore this UI challenge. Phlat combines keyword and property-value search in a seamless and intuitive manner, allowing users to find information based on whatever they may remember, wherever that information may be stored. In addition, Phlat provides a facility for tagging content with user-created metadata to insure that users can return to their content.

Personal desktop search systems available today are remarkably powerful. They unify a number of disparate silos of information (e.g., file system volumes, email, Web histories, etc.), such that a search from a single interface may span many locations and information types. In addition, these systems are able to index a range of structured metadata beyond the content of stored items, including dates, authors, bitrates, etc. Phlat was created to leverage these powerful features of search systems so that users can reliably and intuitively find their content.

RELATED WORK

Since Vannevar Bush first described the *memex* [2], "…a device in which an individual stores all his books, records, and communications, and which … may be consulted with exceeding speed and flexibility," computer scientists and system designers have worked to make his vision manifest. We believe that the materials for Bush's vision are available now, but there is still much to do. The volume of personal information is growing exponentially while our brains stay the same. Personal Information Management (PIM) is a rapidly growing area of research concerned with how people handle this avalanche of information [12].

In 1983, Malone observed workers in offices organize paper material and then suggested how computer systems might support similar activities for electronic content [16]. Following this, Lansdale described many of the psychological challenges associated with PIM [14]. He characterized two key problems that any PIM tool must address: *First, categorizing items is cognitively hard.* Much of the difficulty associated with finding and managing personal information stems from interacting with filing systems and category structures. These problems occur first in setting up appropriate structures and deciding how to file the content; and then, when it's time to retrieve information, deciding how it might have been categorized. *Second, people remember far more about items than can be used by retrieval procedures*. If the content in people's memories could be used by retrieval systems, we would truly have an "external memory."

Most of the research in PIM has focused on the first of these issues: making categorization easier, more useful and less painful [5, 11, 13, 18, 20]. In contrast, there has been less effort applied to the second of Lansdale's challenges: enabling retrieval systems to utilize whatever people remember about their content. Lifestreams [8] enables searching for personal content with a particular emphasis on time and the visual nature of electronic documents. Memory Landmarks [19] also focuses on time, but with special attention to personal and public events. MyLifeBits [9] and Stuff I've Seen (SIS) [7] both provide content search and filtering for various metadata properties, but these systems are focused more on the system infrastructure and functionality than the user interface. Although not focused on personal information, Flamenco [28] explores UI for flexible search on a variety of orthogonal properties (facets) to provide rapid exploration and query iteration based on the interests of the user. Finally, several of the commercial products noted earlier provide search and filtering on properties that go well beyond simple keyword search for content. However, we believe that much more can be done to truly meet Lansdale's second challenge.

Human memory is highly dependent on the relationship between encoded content and the cues and context associated with that content [4, 25]. This means that users' memory for information is highly dependent on lots of other information *not directly associated with what they're looking for*. Czerwinski and Horvitz [3] showed that people forgot a great deal about their computing tasks after just one month. But when prompted by videos and photos of their work during that time, they remembered many details about what they had been doing. Task-centered management tools such as UMEA [13] and TaskMaster [1] address this to some degree by capturing the task context of content. But often the associative threads connecting pieces of information are impossible for a system to infer and probably only make sense to the user himself. Anything that makes these associations easier to explore will make information easier to find.

We created Phlat with the desire to build and deploy an interface that would make finding personal information easy and intuitive no matter what users may remember about their content. We believe that personal search will be every bit as important in the evolution of PIM as internet search has been for the World Wide Web.

DESIGN PRINCIPLES

We approached the design of Phlat with several key principles. First and foremost we wanted to design an interface that stretched the idea of what "search" is. We believe that the dichotomy of *search* and *browse* is an artificial distinction [15]. In Phlat we wanted to facilitate all information seeking as free-text and/or structured property search. Rather than viewing *search* and *browse* as separate behaviors, Phlat treats them as two ends of a smooth continuum. In addition, while we believe that a good search system will dramatically reduce the need for organizational structures such as folders, there are clearly times when users need to apply external labels to their content. To support this need, we wanted to design a convenient and intuitive mechanism for assigning and filtering on user-generated metadata (tags). We felt the following design points were critical to achieving these goals:

1) **Unify text entry and filtering.** If nothing else, the enormous success of Web search has shown that people are very comfortable searching for a wide variety of information with just a few words of text. Phlat must encourage free text entry and iteration as a core part of the interface. But in addition, we want to make it clear that a filter *is* a query. Therefore a user should be able to initiate a query by applying a filter, typing text, or both, in any order and the query representation should reflect this unity of free-text and structured search.

2) **Current search criteria must be visible and salient at all times.** At any moment, a user must be able to glance

at the interface and know exactly what the current search criteria are, what filters are in place, what search terms have been entered and what the current sort order is.

3) **Provide rapid query iteration.** When a user hits *Enter*, clicks a filter button or otherwise changes the current query, results must appear quickly and the change must be obvious. We take a cue from Shneiderman's dynamic query interfaces, which provide continuous visual updates of the results set based on the manipulation of data attributes [21, 22]. Query previews [17] couple data counts with attributes to improve the chances that a query manipulation (e.g., a filter) will return a useful set of results—neither too huge nor too small. We want users to be able to explore their personal information in a natural and fluid manner.

4) **Allow iteration based on recognition.** Because recognition memory is generally much more robust than recall, Phlat should exploit this fact to its users' advantage. When a user views a set of results and properties, he may see something that points him to his information goal. Phlat should enable him to use whatever information he sees to further direct his search.

5) **Allow for abstraction across property values.** Because Phlat is designed to search across a wide range of types and sources, it is important to support a number of property abstractions to match the cognitive models of our users. A user should not have to remember whether the image he's looking for is a bmp, jpg, png, etc.. He need only click on the "Picture" filter and Phlat will query for all of them.

6) **Tag UI must support both tagging and filtering.** Users do not want to have separate UI for application of tags and filtering. There should be a place that users go to interact with tags and this should support both filtering (to find things labeled with a tag) and applying tags to personal information.

7) **Integrate with common file system/mail operations.** Because we expect users to employ Phlat to interact with their content across a variety of sources, users should be able to apply common operations such as cut, copy, paste, drag and drop, reply, etc. from within the interface.

PHLAT

General architecture

The Phlat application was designed as a shell for interacting with personal information on the client machine. Phlat is written in Microsoft Visual C# and uses the Windows Desktop Search indexing and search engine. Windows DS is a free application associated with the MSN Toolbar. For details of the system architecture of Windows DS, see [27].

Phlat provides access to content in the Windows DS index. By default, Windows DS indexes the user's email and personal files including media. Other sources can be added, including the Web cache and shared network directories.

In addition, Phlat allows users to add hierarchical user-created metadata, or *tags* to displayed content. While applying tags following a search is clearly not the only (or best) time to provide this functionality, we felt it was a reasonable first step for exploring tagging UI.

We considered two ways for implementing tags: we could either implement a separate database of tags or apply the tags directly to stored items. We chose the latter for several reasons: First, it allows a single tagging system to span multiple sources (email, appointments, files in different physical or logical locations). Second, Windows DS is able to directly index the tags, allowing Phlat to treat queries for tags just like any other property query. Third, because tags are directly associated with the files, the work of tagging is much safer. If something happens to the index, the metadata is still safely associated with the items. Fourth, items need only be tagged once: when I copy a file to another system, the tag travels with it. Phlat implements tagging by inserting information into a custom metadata property associated with the target item(s). For email items, this is a MAPI property and for files (and Web cache items), this is an NTFS property. A limitation implied by this architecture is that tagging for mail only works for email in Microsoft Outlook/Exchange that supports MAPI. Tagging for files only works on systems supporting the NTFS file system.

User Interface

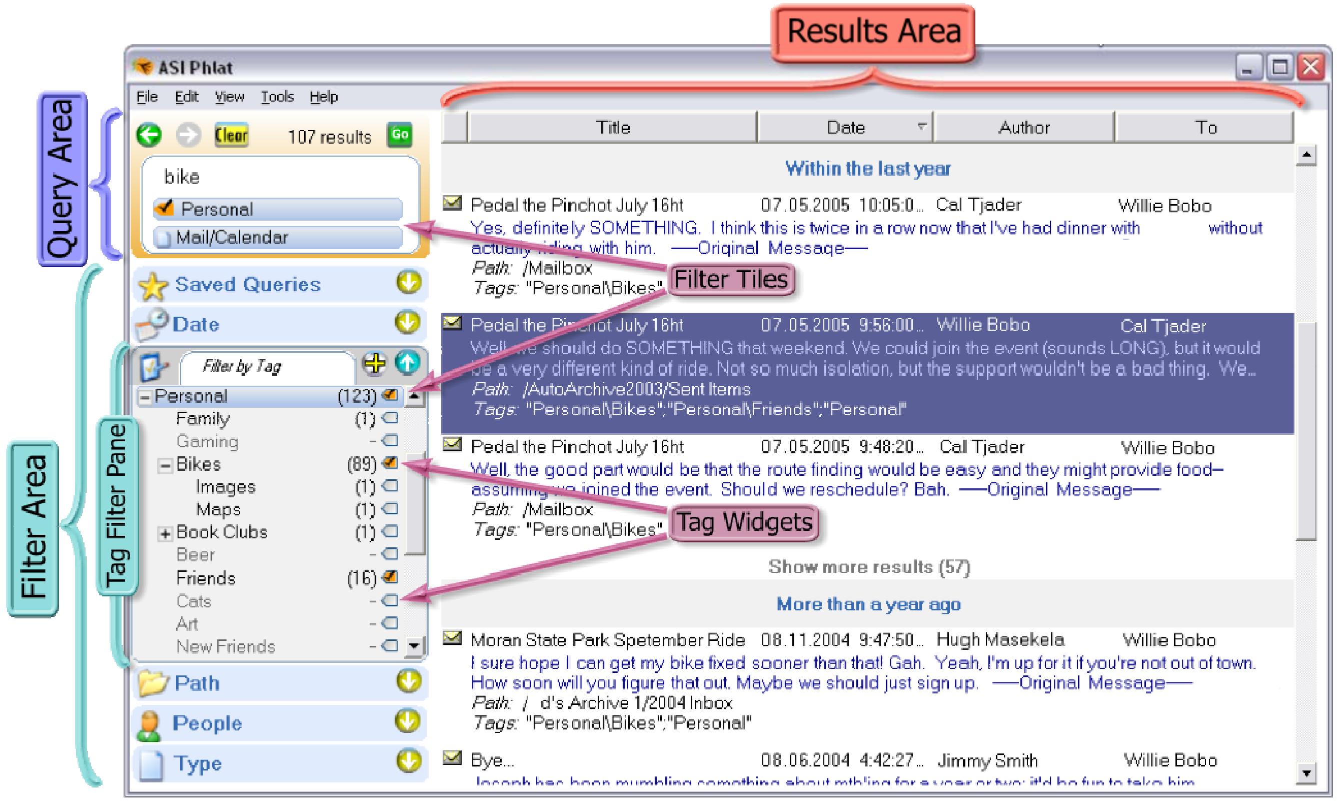
The Phlat user interface is composed of 3 main areas (see Figure 2). In the upper left corner is the Query Area, comprising query controls (e.g., Go, Back, Forward, Clear/ Stop) and a query box for entering and displaying queries. Below the query area is the Filter Area which comprises a set of buttons for several orthogonal properties. Finally, to the right of the Query and Filter Areas is the Results Area. Each of these areas is described in more detail below.

Query Area

The Query Area is the primary space for query control and status, reflecting the current query (text typed by the user and/or any filters) and the number of matching search results. The query box automatically expands downward as additional lines are needed for typed query terms or filters (see below). Typed query text is always kept at the top of the query box and filter tiles are added below this.

Each time a property filter is selected, a tile representing that filter is added to the query box immediately below the typed text. Subsequent filter tiles are place above this. The only exception is that tiles for multiple values of a property are grouped (e.g., in Figure 2 additional Type filter tiles would be placed below the "Personal" Tag tile, but above "Mail/Calendar"). If there is no typed text, we reserve a single blank line at the top to invite further query text.

Consistent with most commercial Web search engines, all typed query terms and filters are joined by an implicit AND (this can be overridden by query syntax). To reinforce this notion in the UI, each typed query word is entered as its



Pedal the Pinchot July 16ht 07.05.2005 10:05:0... Cal Tjader Willie Bobo
Yes, definitely SOMETHING. I think this is twice in a row now that I've had dinner with without actually riding with him. —Original Message—
Path: /Mailbox
Tags: "Personal\Bikes"

Pedal the Pinchot July 16ht 07.05.2005 9:56:00... Willie Bobo Cal Tjader
Well, we should do SOMETHING that weekend. We could join the event (sounds LONG), but it would be a very different kind of ride. Not so much isolation, but the support wouldn't be a bad thing. We...
Path: /AutoArchive2003/Sent Items
Tags: "Personal\Bikes";"Personal\Friends";"Personal"

Pedal the Pinchot July 16ht 07.05.2005 9:48:20... Cal Tjader Willie Bobo
Well, the good part would be that the route finding would be easy and they might provide food–assuming we joined the event. Should we reschedule? Bah. —Original Message—
Path: /Mailbox
Tags: "Personal\Bikes"

Moran State Park Spetember Ride 08.11.2004 9:47:50... Hugh Masekela Willie Bobo
I sure hope I can get my bike fixed sooner than that! Gah. Yeah, I'm up for it if you're not out of town. How soon will you figure that out. Maybe we should just sign up. —Original Message—
Path: / d's Archive 1/2004 Inbox
Tags: "Personal\Bikes";"Personal"

Bye... 08.06.2004 4:42:27... Jimmy Smith Willie Bobo

Figure 2. The Phlat interface with a query of a single keyword and two filters.

own line just as each filter tile gets its own line in the query box. The only exception to this rule is when the query is a phrase in quotes (e.g., *"information retrieval"*). Then, the entire phrase is placed on a single line, wrapping with a slight indent if needed. So, for example if a user typed *intuitive "information retrieval,"* Phlat would render *intuitive* on one line and *"information retrieval"* on the second. This convention reinforces to the user that each line is an independent element in building the query.

The integration of property filter tiles into the query box serves several functions. Rather than treat filters as a form of navigation orthogonal to typed query terms, filters are explicitly part of the query. Integration in the query box reinforces that filters are just a structured query with property restrictions.

Second, by integrating filters into the query box, we are largely able to mitigate the "stuck filter" problem endemic to many search interfaces. This problem is illustrated by a user who searches for a document using a few keywords and a date filter. A bit later he returns to the search interface to find another document, enters new keywords and cannot find what he's looking for because he's forgotten that the date filter is still in place. This scenario is quite rare in Phlat because all filters are so closely associated with other query terms. It's very hard not to notice that a filter is in place when you must type a new query adjacent to it!

Finally, the query box is the central locus for query refinement. To remove or edit a filter, simply clicking the tile in the query box is sufficient. A right-click context menu is also available for changing the filter into an exclusion (e.g., the Calendar filter would change into NOT Calendar). The filter would then change from blue to red to reflect this change.

Filter Area

Currently, the filter area is made up of six buttons: Saved Queries, Date, Tags, Path, People, and Type. When a user clicks on a button, it opens, revealing a pane that expands to fill the available space below the Query Area. Each filter pane lists the values associated with the property that can be used to filter the current results (e.g., Tags in Figure 2). As more property panes are opened, they split the remaining space with the constraint that all buttons in the filter area must be visible and actionable at all times.

When a property value within a filter pane is clicked, it is highlighted as a filter tile and a corresponding filter tile is added to the query box as described above (e.g., the "Personal" Tag filter in Figure 2). Clicking on a tile in either area will remove the filter from the query box and turn off the highlighting in the filter pane. This makes it very easy for users to iterate on a query regardless of whether their attention is located in the Filter Area or the Query Area.

For most properties, the values displayed are only those which can filter the current results; if there are no multimedia files in the current result set, the Type filter will

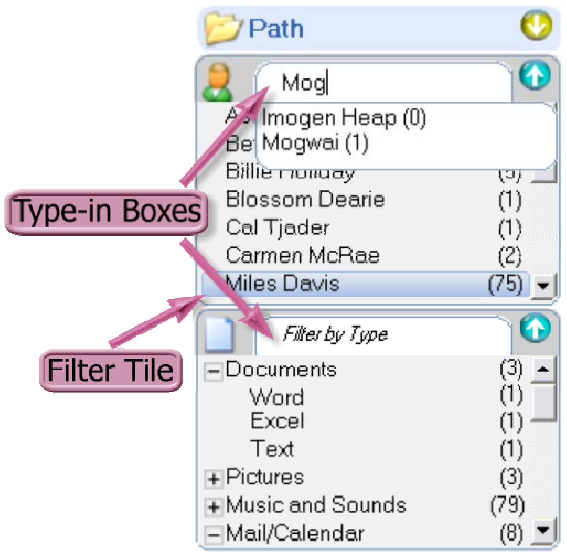


Figure 3. Path, People and Type filter panels with a partial word in the type-in box for People with matching values in drop-down box.

not list any music or video file types because filtering on these would yield an empty result set. In addition, next to each property value is the number of matching results for that filter value. In Figure 3, when the user clicked on “Miles Davis,” 75 results were returned. This information helps users better predict whether a given filter will be useful in narrowing their results appropriately [17].

As noted earlier, a key design principle for Phlat was to provide multiple levels of abstraction for properties. Property filters may include different kinds of abstractions. People filters are an interesting example. While users can (and do) simply type the name of a person into the query box, this is often too general, because it will return all results which have that word *anywhere* in the content. If a user is looking for a mail thread with Bill, he would probably like to avoid all the documents about the Bill of Rights. However, he may not remember whether Bill was on the From: To: or Cc: line of the mail. The People filter accommodates this by querying across only the people fields in the index (e.g., MailTo, MailCc, Author, Artist, etc.). For Date, Phlat queries a special column in Windows DS called “PrimaryDate” which is an abstraction of user-relevant dates for different file types. For email, the date received is used; for Web pages, the access time; for calendar items, the date of the event; and for photos, the time the photo was taken is used. Finally, the Type filters are arranged in a shallow hierarchy that includes a number of type abstractions such as Documents (which includes office docs, PDF, etc.), Music, Pictures, and Mail/Calendar.

In addition to querying on known property values, Phlat allows users to search for arbitrary values. Once a property filter pane is opened by clicking on it, the title for that pane is replaced with a small type-in box with the invitation text, “Filter by xx” where xx= the property for that pane (e.g., Type in Figure 3). When a user begins typing, a drop-down appears below the box containing all property value matches following each keystroke (after only 3 keystrokes, the set of matching values is often effectively reduced as in Figure 3). If a user selects a value from the dropdown, that value is highlighted in the property pane and added to the query box just as if they had clicked on the value. However, if the user simply hits *Enter* without selecting a specific value from the dropdown, a new filter tile with the typed text is added to the query pane and the current results are filtered accordingly (in Figure 3, a People filter for the word, “Mog.")

This flexible level of specificity is useful in supporting different information needs. Let’s say that a user is looking for a document authored by “George” but he can’t remember the last name. If he types “George” in the People pane, Phlat will return all documents with the word “George” in a people property (Author, MailTo, etc.), including George Orwell, George Furnas, Emily George, etc. Combined with other things he knows about the document, he may then find *Middlemarch* by George Eliot in the results set.

The Saved Queries pane is different from the other five filter panes in that its members are not actually filters. Any query (combination of keywords, filters and sort order) can be stored as a saved query. When a saved query is clicked, the current query is replaced by a query tile representing the saved query. Users are then free to iterate on the query however they choose.

Tagging

The design of the Tags pane posed a special challenge. We conducted several user studies on tagging and filtering designs, including a high-fidelity Flash prototype and a (different) pilot system. From these tests, we knew that users expected tag application and filtering to occur in the same place. This means that the same UI has to support both retrieval and organizational tasks. Filtering for tags needs to function like all other filters to provide a consistent search experience. However, users must also be able to easily apply these same tags to their documents. All this functionality can quickly clutter the UI. One solution for tagging is to support drag and drop (“drag-to-tag”). A user can tag an item in the results by either dragging and dropping a tag onto a result or dragging and dropping a result onto a tag. While this works quite well and is somewhat intuitive, it is not very discoverable and by itself provides little feedback that changes have been made. In addition to drag-to-tag, we also created a tag widget (see Figure 2). This UI element is a small checkbox widget at the right edge of the Tags pane. To reduce clutter, the tags widget remains hidden while users interact with the Query or Filter Areas, but the widgets appear whenever one or more items in the results pane are selected. At this point a user can quickly and easily apply a number of tags to the selected result(s) by checking all the appropriate tags.

Similarly, by selecting a given result, a user can see what tags are associated with it at a glance (in Figure 2, the selected item has 3 tags).

Results Area

The Results Area displays the search results in a columnar list view. Property headers are listed across the top and the corresponding properties in the results appear in columns below this. A variety of fields other than those shown in Figure 2 are available for display through an options menu (e.g., Mail CC, Relevance Score, Index Date, Genre, etc.). Following the findings for the SIS system [7], the default sort order for search results is by date. Clicking on a column header sorts by that column, maintaining a stable sort (i.e., clicking on the Date followed by Author column headers would sort the search results first by Author then by Date). Search results are grouped by the primary sort column. If there are more than 10 results in a given group, additional search results for that group are collapsed and the next group is displayed beneath it (see Figure 2). The grouping functionality can be disabled by users.

As with the filters, several property abstractions are used for displaying results. The default date displayed is PrimaryDate (described above as the relevant date for a given item type). Similarly, the value appearing in the Title column depends on the document type. For email, this is the Subject; for music, the song title; and for Web pages, we display the page title. Even though many file system documents (PDF, Word, PowerPoint, etc.) contain metadata fields for document title, we found that users prefer to see the filename rather than the document title. This is because document titles are often incorrect, misleading or simply empty for many files. Since users typically name their own files, it is the filename that they recognize rather than the document title.

Below each search result, we optionally display a snippet of the first few words in the search result item, the folder path (mail, file system, shared folder, etc.), and any tags that may be associated with that item. Users can turn this additional information off to maximize the number of results displayed.

Users can interact with results in much the same way they can in Windows Explorer or Outlook, e.g., drag and drop, open, delete, etc. In addition, a popular extension unique to the search environment is the capability to open the parent folder of a given result (also in OS X Spotlight). This allows users to quickly navigate to a folder based on a single result item. For example, a user may be trying to locate a set of documents associated with a project. If they find any one of those documents in Phlat, they can easily get to the folder containing that item to find the required documents.

The results can also be used to narrow or replace the current query. Right-clicking on any visible property in a result (e.g., the document title, “Pedal the Pinchot July 16ht” in Figure 2) brings up a context menu that allows the user to either add a filter for that property value to the current query, or *replace* the current query with a query restricted to that property value. For example, we could either add a filter for that document title to the current query, or clear the old query and issue a new one for all items in the index titled “Pedal the Pinchot July 16ht.” This allows users to “move sideways” in their searching, based on information they may recognize in the result set.

PHLAT USAGE AND EVALUATION

We report quantitative and qualitative data from 225 people who used Phlat during an 8 month period from January 25, 2005, through August 31, 2005. Phlat was installed by a large number of employee volunteers in our organization as a prototype search application. Our users spanned a wide variety of jobs, including program management, development, sales, administration, managers, and executives.

It is extremely difficult to study systems for personal information retrieval in the laboratory for a variety of reasons. Because these systems are specifically designed to leverage a user’s memory and knowledge of his own content, any test must employ that content. This makes task creation very difficult because each task must be tailored to each individual user and is associated with a variety of privacy problems. We have done some work in the lab studying specific UI designs for Phlat particularly using eye tracking techniques (forthcoming). In this paper we summarize our observations of how Phlat is used in the “wild” as people incorporate it into their daily lives.

Here we describe analyses of detailed usage logs and spontaneous feedback by our users. Research using query logs for Web search has yielded important information about how users interact with such systems [23, 24]. Log analyses have also proven very useful for understanding other systems for personal search [7].

Analysis of Usage Logs

Phlat was instrumented to capture many user interactions with the interface. Example data included query text, filter usage and tagging activity. In addition, we gathered information about items opened through Phlat, such as age, type and storage location. For privacy reasons, we did not log any information about the content of search results or users’ indices.

The size of the personal indices of this sample of users varied by almost 2 orders of magnitude, ranging from 5733 items to more than 472000 items. The median index size was 36182 items.

Sessions

To better understand our data, we divided the usage logs into *sessions*. A session comprises a series of related queries by a single user over a small period of time. This is meant to capture a given user’s attempt to fill a single information need. We assume that queries for a given information need should come clustered in time and should

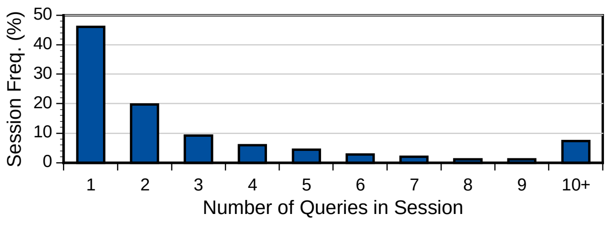


Figure 4. Distribution of session frequency and the number of queries per session.

be semantically related. Following Silverstein [23], we defined a session as all queries with an inter-query interval of less than 5 minutes. When the inter-query interval was longer than 5 minutes we checked the semantic content of the subsequent query by comparing the query terms. If it was different, that query started a new session.

During the time period studied, users issued more than 18500 queries in 5144 sessions, for a mean of 3.61 queries per session (Figure 4).This is almost twice the 2.02 queries per session reported by Silverstein for the Web. This is probably due to the very easy query iteration Phlat makes possible; users cast a wide net and then narrow their results.

Query Characteristics

Queries were generally very short, averaging 1.60 words (stddev=1.31). This is somewhat shorter than the 2.35 reported for the Web [23], but almost exactly the same as the 1.59 words reported for the SIS system [7]. As in SIS, short queries are quite effective because it is very easy to quickly iterate by filtering and personal stores are smaller and more familiar to users than the Web. Explicit Boolean operators (AND, OR, NOT, +, -) and phrase restrictions were very rare (<1%). Interestingly, even with the very easy filtering offered by the Phlat UI, typed filters in the query box (e.g., *from:Bill*) occurred in 6.4% of all queries.

The Phlat interface was designed to encourage the easy use of filters (i.e., field restrictions) through the use of the dedicated filter widgets. The user logs suggest the design was successful. Forty-seven percent of all queries involved some kind of filter. The most common filter employed was for People, followed closely by file Type (see Table 1). The very strong usage of people filters is also consistent with the report by Dumais et al. that 25% of all free-text queries in the SIS interface included people's names[7].

Filter Type	Number of Queries
Tag	1297 (7.0%)
Path	1354 (7.3%)
Date	1425 (7.7%)
Type	3101 (16.7%)
People	3401 (18.3%)

Table 1. Frequency of filter usage in the Phlat interface.

When users employed filters, they often combined several together. Indeed, more than a third of all queries with filters had more than one filter, and it was not uncommon to see more than 5 in a query. Interestingly, users often issued queries using *only* filters (i.e., 17% of all queries comprised a filter with no query text in place). For instance, a user might click the *Today* filter followed by a *People* filter to find all the interactions with a given person that day.

Finally, because of the wide range in the size of users' indices, the relatively short queries and the tendency to iterate, return sets of search results varied widely from 0 to more than 225000 hits, with an average of 478 results (stddev=3184).

Search Results

The true measure of the usefulness of a search UI is whether it gets you useful results so you can carry on your tasks! In the more than 5000 sessions analyzed, almost 13000 search results were *invoked* in some fashion. That is, the items were opened, copied, dragged, deleted or otherwise used in some way.

On average, 2.5 items were invoked per session (stddev=5.24). However, in about 25% of all sessions, no search result was invoked. These sessions may represent failed searches, but it is also possible that users employed the search results in other ways. For example, the information they needed may have been in the snippet or other metadata displayed in the search results so there was no need to actually invoke the item. A good example is a search for a phone number in an email. If the number is in the snippet, there is no reason to bother opening the item.

For each invoked item, we recorded the type, date, and rank position in the result set. Similar to the SIS results and reflecting the work habits of enterprise users, email and appointments in Microsoft Outlook were by far the most common type opened (80%) followed by Microsoft Word (6%), Microsoft Powerpoint (3%), and various media types including pictures and music (4%). One interesting finding was the large number of email attachments that were invoked using Phlat. Over 6% of all items were file attachments of various sorts (Word files, spreadsheets, PDF files, etc.). This echoes the observations of Ducheneaut and Bellotti [6] that email is widely used as a file repository and organizational tool. Indeed, many users treated Phlat primarily as a tool for searching their email archives.

We also looked at the rank position of each invoked item. Figure 5 shows the frequency of invoked items as a function of rank position in the Phlat result list. It is interesting to compare and contrast these results to similar data for Web search. Unlike Web search which is almost always sorted by a relevance score, Phlat search results are usually sorted by metadata that is very meaningful to users (by default, Phlat sorts by Date, but Author, MailTo, Size, and others are commonly used). As a result, one might expect the rank position of results in Phlat to be less important because the relative ordering of the metadata provides sufficient orientation to get to desired information. In addition, Web search usually returns about 10 results per

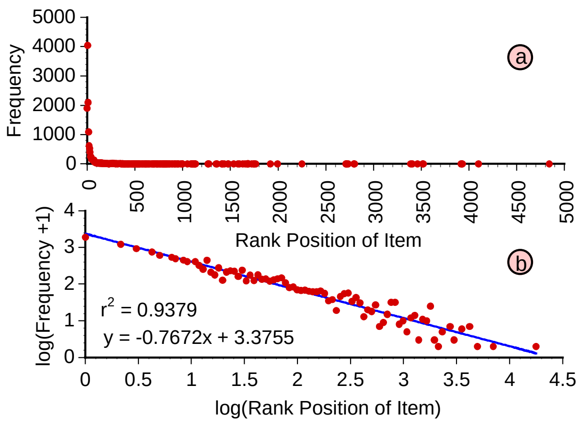


Figure 5. Frequency of invocation of results in Phlat.

page and more results require additional page views. In Web search, the top 3 positions account for more than 70% of all items opened [10]. As we expected, this is less true for personal search using Phlat. In Phlat, only 30% of all invoked items are in the top 3 positions, and there is an extremely long tail (see Figure 5a). So for example, if a user is looking for a very old item, he has a good idea as to relative position of the target in the current result set. Rather than re-query to bring the item closer to the top (and run the risk of over-specifying his query), he can quickly scroll to the end of the result set to find the information.

In Figure 5b we plot the access position on a log-log scale. The linear fit is excellent ($r^2 = 0.94$, $p<<0.0001$). This figure suggests that most users do select items that are ranked near the top, regardless of how they are sorted, but are willing to scroll well into the set if needed. Since results are sorted by a metadata property that users understand, the navigation to target information is straightforward.

Finally, we make special note of the extensive use of filtering on tags (15% of all queries using filters). This number is quite impressive, given that users had to tag their content themselves, and the only facility for doing this was in Phlat. About a third of our users experimented with tagging at some point, and of these, 12 users (including technical, executive & sales persons) consistently used tags for the duration of the study. For these users, tagging seems to be central to their use of Phlat.

User Feedback and Usage Observation

When we deployed Phlat we created a feedback mailing alias in which users could seek help, report bugs and make comments on their experience using the prototype. We explicitly solicited user feedback so that we could better understand how our design was meeting, and just as importantly *failing to meet*, the needs of our users.

We have received hundreds of emails from our users concerning Phlat. Communications can generally be classified as bug reports, functionality requests, complaints, and complements. Below, we focus on the strengths of Phlat, our most common functionality requests, and finally problems with our implementation.

Raves

People appreciate the simple but expressive query model and quickly begin to issue what in other interfaces would be very complex queries. Our users are particularly enthusiastic about the idea of cross-source searching and tagging. We repeatedly heard users delight that they didn't need to worry about *where* an item was stored, and many loved the idea of creating a single organizational structure (tags) that spanned email and files (though, as noted above, only a small number of users regularly employed tags).

Another feature that was particularly lauded by users was the ability to modify their queries based on information in the search results (i.e., right-clicking on a value in the results to modify or replace the query). Although this feature was not very discoverable, once users knew about it, they appreciated the ability to search laterally based on their current results.

Perhaps the best evidence that our users appreciate Phlat is their continued use. Because Phlat is simply a shell running on top of Windows Desktop Search, much of the base utility of personal search is available without our tool. The fact that our users continue to employ Phlat in their daily work indicates that they are receiving additional benefits from using Phlat.

Desires

Our users supplied us with many and varied functionality requests. Some we were able to quickly implement, but many we were just not able to accommodate.

One of the single most requested features was for a preview viewer for search results and the display of fast thumbnail views for search items. Indeed, the preview viewer associated with Windows DS was probably the single most common reason for users to stop using Phlat in favor of the default interface for Windows DS.

Another common request was for increased integration of Phlat functionality within Microsoft Outlook and the Windows OS shell. Users wanted to be able to launch searches from within any application with a simple context menu. In addition, users wanted tagging integrated into other applications; they wanted to be able to tag items from within Outlook, Windows, and other apps. Microsoft Outlook already provides tagging of keywords and flagging functionality; a number of users wanted to be able to unify this existing functionality with Phlat. Along with some issues noted below, this was a major barrier to the continued usage of tags.

Finally, a third major request was related more to the underlying index than Phlat *per se*, but we report it here because it speaks to the kinds of information needs users try to satisfy in personal search. A number of users complained that Phlat would often return the correct file for their information need, but the granularity was too broad to help them. This commonly occurred for two main file types: Microsoft PowerPoint and Microsoft OneNote. The issue here is that both PowerPoint and OneNote files are monolithic files comprising a number of smaller units. However, the granularity of the indexing system is at the file level rather than at the slide or notebook page level. As a result, a user may open a file only to be confronted with yet another search within the document itself. This could be addressed with either a different level of indexing or with an improved ability to navigate within large documents.

Problems with Phlat

While Phlat has been very successful on a number of fronts, there remain several real problems to be confronted. Here we discuss three areas for improvement.

First, there are several problems associated with the fact that Phlat runs on top of a search engine that is independent of the file and mail systems on a client machine. Changes to objects take some finite amount of time to propagate to the index, so new email or files may not be immediately available for searching. This time can be fairly substantial following changes to a large number of items, and can lead to inconsistent search results. For instance, it may take minutes to update the index following the trivial act of moving a folder from “Work” to “Done.” In the mean time, these items will not be accessible because the path has changed. Since to Phlat, “Path” is just another piece of metadata, the same problem exists for *any* metadata change, including tagging changes. Until the index is updated, filtering on the new metadata will not give any results!

Another problem in Phlat concerns mapping the many representations of people onto one underlying “person” construct. This is a serious problem for many different systems, but is very noticeable in Phlat. Because Phlat simply iterates through the different “people fields,” it is not unusual to see a single person represented as several different filters. E.g., Puffy Combs, Puff Daddy, P Diddy, Sean Combs, puffy@hotmail.com, etc. are treated as different people filters. This can be quite confusing for users because it is then incumbent upon them to figure out which version of the person they need to use to find the object they are looking for.

Finally, there are two major problems with the current implementation of tagging. The first is a limitation associated with how we apply the metadata associated with tags. Because tags are applied as custom metadata properties in MAPI (for mail) and NTFS (for files), there are a variety of ways in which they can be lost. For instance, tags do not travel with forwarded or replied mail, so subsequent messages will not inherit the tag. Tags are also lost if a user drags a mail message onto their desktop. For files, if a user copies a file onto any non-NTFS file system (such as a CD, DVD, or USB flash memory unit), all tags are removed. This is clearly a serious hurdle. Classification and tagging require serious cognitive effort [11, 14], and if the metadata of tags is too fragile, users will stop spending the time and energy to use them!

The second problem with tagging in Phlat is that tagging is *only* available through Phlat. This means that the only time that a user is able to apply tags to content is at retrieval time. However, much of the benefit of tagging is to aid retrieval in the first place! Currently Phlat supports “spring cleaning,” that is, when users are going through their content *en mass* and organizing a number of items at the same time. Phlat also supports incidental tagging for future use (e.g., a user finds some information after a search and wants to make sure they can easily find it in the future). While tagging in Phlat *is* useful for a core group of users, its utility would be vastly increased if users were able to tag their content any time they interacted with it. In particular, there are certain key “inflection points” that are critical. For mail, this is during “mail triage” [26] and for files it is during the “Save dialog” when users are thinking about how the document will be recovered later. And for all documents and web pages, the system must support tagging during work flow as users encounter new things and incorporate them into their daily tasks.

CONCLUSIONS AND NEXT STEPS

We designed and deployed a system for quickly and intuitively searching for and organizing personal information. Feedback and usage information suggests that we succeeded in building a system that assists users in finding their personal information by allowing them to search on a wide variety of features based on contextual information and associative cues in their content. In addition, our experience with tagging points to several important insights in design and functionality. Despite its many limitations, our tagging system is used quite extensively by a subset of users. Clearly, our users find the idea of universal tagging compelling.

Our experiences with tagging have strong implications for the development of future systems. As operating systems move to incorporate fast search as a core functionality, user-generated metadata will become increasingly critical to how users organize their content. Filing will become less important and will be replaced by more general tagging systems. However, if these systems are not supported throughout users’ workflow, we may find that users are not much better off than they are today.

A future with ubiquitous personal tagging raises a plethora of issues relating to sharing and privacy. Web-based systems such as Flickr (flickr.com) and del.icio.us hint at the power of tagging in shared scenarios, but these are mainly public systems. How do we extend this power to accommodate personal information? A second question for further investigation concerns the use of hierarchy vs. flat organization of tags. Flat systems are easy to understand and lend themselves particularly well to shared scenarios (such as the Web systems above). However, they do not scale very well; searching a list of even a few hundred tags can be an onerous task. In contrast, hierarchical systems offer a great deal of organizational power, but are very hard to use in shared-author settings. Root-leaf relationships are simply not always obvious or consistent and may change depending on the context of use!

Finally, we continue to explore the domain of rich search interfaces, an important question to be answered is whether these designs can be extended to include "non-personal" content. For instance we would like to see whether we can include information sources of interest that users aren't familiar with (news, intranet, etc.). How well does this type of UI support search when there is little or no contextual and associational memory for the information needed?

ACKNOWLEDGMENTS

The authors thank Eric Horvitz, Ken Hinckley & Gina Venolia for great discussions on Phlat's design and feedback on this manuscript. We also thank all our patient and articulate users!

REFERENCES

1. Bellotti, V., Ducheneaut, N., Howard, M. and Smith, I. (2003). Taking email to task: the design and evaluation of a task management centered email tool. In *Proc. SIGCHI 2003, 5*, 1, 345-352.
2. Bush,V. (1945). As We May Think. *The Atlantic Monthly, 176*, 101-108.
3. Czerwinski, M., and Horvitz, E. (2002). An investigation of memory for daily computing events. *Proc. of HCI 2002*, 230-245.
4. Davies, G. and Thomson, D., eds. (1988). Memory in Context: Context in Memory. Wiley: England.
5. Dourish, P., Edwards, W. K., LaMarca, A. and Salisbury, M. (1999). Presto: An experimental architecture for fluid interactive document spaces. *ACM Trans on Computer-Human Interaction, 6(2),* 133-161.
6. Ducheneaut, N.; Bellotti, V. (2001). Email as habitat: An exploration of embedded personal information management. *ACM Interactions*. Sept.-Oct.; 30-38.
7. Dumais, S.T., Cutrell, E., Cadiz, J.J., Jancke, G., Sarin, R. and Robbins, D.C. (2003). Stuff I've Seen: A system for personal information retrieval and re-use. In *Proc. SIGIR 2003*, 72-79.
8. Fertig, S., Freeman, E. and Gelernter, D. (1996). Lifestreams: An alternative to the desktop metaphor. *Proc. SIGCHI 1996*, 410-411.
9. Gemmell, J., Bell, G., Lueder, R., Drucker, S. and Wong, C. (2002). MyLifeBits: Fulfilling the Memex vision. *Proc. of ACM Multimedia'02*, 235-238.
10. Joachims, T., Granka, L., Pang, B., Hembrooke, H. and Gay, G. (2005). Accurately interpreting clickthrough data as implicit feedback. In *Proc. SIGIR 2005*, 154-161.
11. Jones, W., Bruce, H., Foxley, A., Munat, C.F. (2005). The Universal Labeler: Plan the project and let your information follow, *Proc. of ASIST 2005, November*.
12. Jones, W. (In Press). Personal information management. *Ann. Rev. Info. Science and Tech.*, in press.
13. Kaptelinin, V. (2003). UMEA: translating interaction histories into project contexts. In *Proc. SIGCHI 2003, 5* 353-360.
14. Lansdale, M. (1988). The psychology of personal information management. *Applied. Ergonomics, 19*, 55-66.
15. Mackinlay, J.D., and Zellweger, P.T. (1995). Browsing vs. search: Can we find a synergy? (panel session) In *Proc. SIGCHI 1995*, 179-180.
16. Malone, T. (1983). How do people organize their desks? Implications for the design of office information systems. *ACM Trans. Office Info. Sys. 1*, 1, 99-112.
17. Plaisant, C, Shneiderman B, Doan, K., and Bruns, T. (1999). Interface and data architecture for query previews in networked information systems. *ACM Trans. Info Sys.*, 17(3):320-341.
18. Quan, D., Bakshi, K., Huynh, D. and Karger, D.R. (2003). User interfaces for supporting multiple categorization. In *Interact 2003—9th IFIP TC13 Intl. Conf. on HCI*, 228-235.
19. Ringel, M., Cutrell, E., Dumais, S., and Horvitz, E. (2003). Milestones in time: The value of landmarks in retrieving information from personal stores. In *Interact 2003—9th IFIP TC13 Intl. Conf. on HCI*, 184-191.
20. Rose, D.E., Mander, R., Oren, T., Poncéleon, D.B. Salomon, B. and Won, Y.Y. (1993). Content awareness in a file system interface: implementing the "pile" metaphor for organizing information. In *Proc. SIGIR 1993*, 260-269.
21. Shneiderman, B., Byrd, D., Croft, B.W. (1998). Sorting out searching: a user-interface framework for text searches. *Communications of the ACM,* 41(4):95-98.
22. Shneiderman, B. (1994). Dynamic queries for visual information seeking. *IEEE Softw. 11*, 6, 70-77.
23. Silverstein, C., Henzinger, M., Marais, H. and Moricz, M. (1998). Analysis of a very large AltaVista query log. *SRC Technical note #1998-14*. On-line at http://gatekeeper.dec.com/pub/DEC/SRC/technical-notes/abstracts/src-tn-1998-014.html .
24. Spink, A., Wolfram, D., Jansen, B. J., and Saracevic, T. (2001). Searching the Web: The public and their queries. *J. Amer. Soc. Info. Science and Tech., 52(3*), 226-234.
25. Tulving, E. and Thomson, D. (1973). Encoding specificity and retrieval processes in episodic memory. *Psychological Review* 80, 352-373.
26. Venolia, G.D., Dabbish, L., Cadiz, J.J. and Gupta, A. (2001). Supporting email workflow. *Microsoft Research Tech. Report MSR-TR-2001-88*. On-line at http://research.microsoft.com/research/pubs/view.aspx?msr_tr_id=MSR-TR-2001-88.
27. Windows Desktop Search Extensibility for Partners (Beta). http://addins.msn.com/devguide.aspx
28. Yee, K.P., Swearingen, K., Li, K., and Hearst, M. (2003). Faceted metadata for image search and browsing. *Proc. SIGCHI 2003,* 401-408.