# What to Do When Search Fails: Finding Information by Association

**Duen Horng Chau, Brad Myers, and Andrew Faulring**
Human Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
{dchau, bam, faulring}@cs.cmu.edu

## ABSTRACT
Sometimes people cannot remember the names or locations of things on their computer, but they can remember what *other* things are associated with them. We created Feldspar, the first system that fully supports this associative retrieval of personal information on the computer. Feldspar's contributions include (1) an intuitive user interface that allows users to find information by *interactively* and *incrementally* specifying *multiple* levels of associations as retrieval queries, such as: "find the *file* from the *person* who I met at an *event* in *May*"; and (2) algorithms for collecting the association information and for providing answers to associative queries in real-time. A user study showed that Feldspar is easy to use, and suggested that it might be faster than conventional browsing and searching for these kinds of retrieval tasks. Feldspar could be an important addition to search and browsing tools.

### Author Keywords
Associative information retrieval, personal information management, user interfaces.

### ACM Classification Keywords
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval. H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles, Graphical user interfaces (GUI); I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques.

## INTRODUCTION
Finding information on your computer can be a difficult task, even if you are equipped with the latest tools. For example, if you do not remember where you have put the item that you are looking for, then you cannot easily navigate to it with browsing tools. And if you also do not remember the name of the item or any text in it, then search tools also do not work. However, you may remember *other* things that go with the item. Indeed, people often recount chains of associations [3, 19], like "I remember receiving a picture from a person who I met at an event that happened last May". Current search tools for the desktop are designed to support *teleporting* – to bring the users directly to their information targets, assuming the users remember keywords about them. The search tools do not support *orienteering* – where the users specify and navigate to their information targets in multiple relatively small steps, as seen in the example above, where each *step* is an *association*. This lack of orienteering support motivated us to create the Feldspar system (see Figure 1), the first tool to fully support multi-step associative retrieval of personal information on the computer.

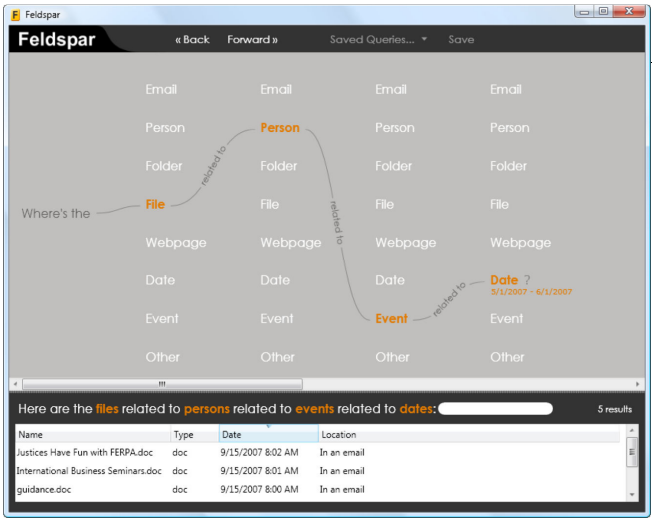


**Figure 1.** The Feldspar user interface showing a user requesting the *file* related to the *person* related to the *event* in *May*

Feldspar stands for **F**inding **E**lements by **L**everaging **D**iverse **S**ources of **P**ertinent **A**ssociative **R**ecollection. Feldspar works incrementally, letting people add one association after another until the desired item is found. At every point, Feldspar presents the results of the query constructed so far, so the desired item can be found with as few query terms as possible. Additionally, Feldspar proposes possibly useful next query terms to add. These techniques can help avoid over-specifying with too many query terms, which can prevent the correct results from being found [1].


*CHI 2008*, April 5–10, 2008, Florence, Italy.

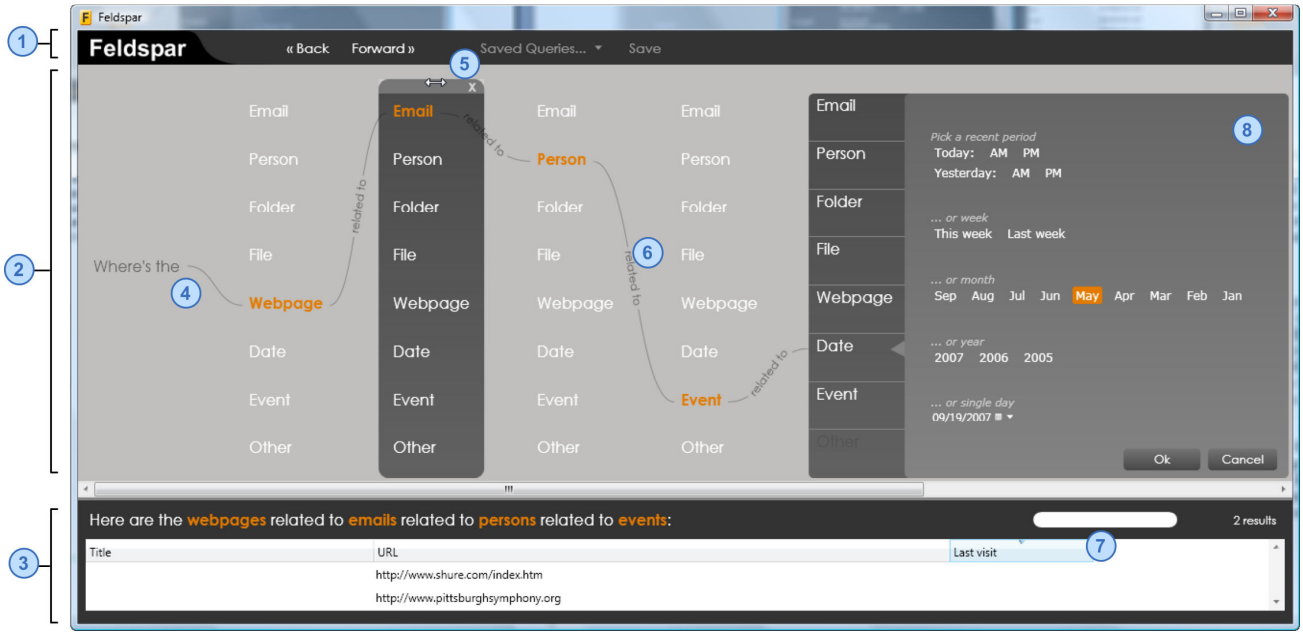


**Figure 2.** The Feldspar user interface. 1: The Navigation Bar. 2: The Query Area for constructing queries. 3: The Results Area with the query represented as a sentence at the top. 4: The main query area. 5: The user can freely edit the type of an association and swap its order with other associations. 6: Items in queries are linked by the term "related to". 7: The user can filter the results by typing a filtering string into the textbox. 8: The date picker panel, which allows the user to pick a data range such as May, or a specific date using a calendar dropdown. At any time, the user can edit the query by selecting different values and the results update immediately.

The important contributions of Feldspar include:

- A user interface that allows users to find information by *interactively* and *incrementally* constructing *multi-level* associative retrieval queries, which has been shown to be usable in a user test.
- Algorithms for collecting the association information and for providing answers to associative queries in real time.

## RELATED WORK

The psychology literature has shown that people often remember things through chains of associations [3, 19]. Our memory works by creating associations between things that we perceive together. When we try to recall one thing, we are reminded of another thing that is *associated* with the first one, which in turn reminds us of yet another, forming a chain of associations. A recent observational study conducted by Teevan, et al. [18] supports this theory. The study shows that people use contextual information associated with their targets to guide their navigation in relatively small steps, as they gradually recall pieces of information associated with the targets. Furthermore, the researchers observed that people actually *prefer* to use this orienteering strategy, even when the teleporting strategy, as exemplified by typical search, would have worked. However, there has been a lack of support for this preferred, and arguably most natural, multi-level associative retrieval strategy, which Feldspar is designed to support.

Many research and commercial systems have been created to support various forms of search and retrieval. For example, desktop search programs such as Google Desktop, Microsoft Windows Desktop Search, and Spotlight for Apple's OS X, help the user find information items by their keywords, assuming the user remembers those words. Dourish, et al. [4] proposed an attributed-based method, similar to tagging, to help people categorize documents. Many systems, such as TimeScape [13] and the Lifestreams system [5] manage files by time, thus supporting time-based retrieval. Ringel, et al. [15] proposed a timeline-based visualization of search results of personal content, showing major events that a user may recognize to help the user retrieve information more easily. Nardi and Barreau [12] suggested the need for better location-based document management systems, after observing that people often placed together documents that share similar types, similar topics, or proximity in creation times. Jones, et al. [6] proposed a project-centric approach for organizing and retrieving electronic documents. Lamming and Newman [8] suggested an activity-based approach to support retrieval by continuously gathering users' activity data and then using the data as contextual cues to assist retrieval. Rhode's wearable Remembrance Agent [14] works similarly: it examines the user's physical context to present information relevant to the context. Some more recent attempts include the commercial tool Xobni (www.xobni.com) and the research system Jourknow [7]. All of the above systems only support one level of association, not the multiple levels that people often remember.

Our earlier Iolite [16] system is the most closely related, but its user interface was never completed. Iolite's focus was on

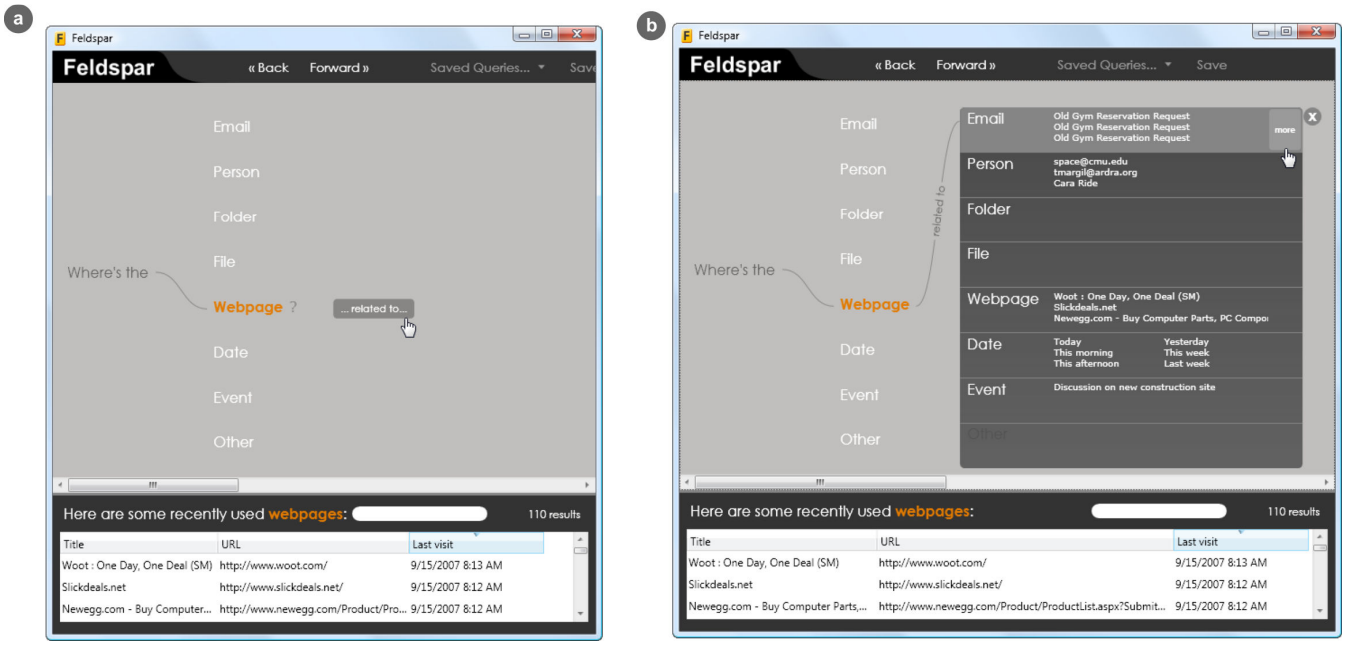


**Figure 3.** (a) Feldspar after the user has selected *Webpage*. (b) Feldspar after user has clicked *"related to"*

designing some of the underlying algorithms, with relatively little emphasis on the interface. The execution of the system was also not fast enough for real-world deployment.

Although faceted search [20] also works incrementally, it is fundamentally different from Feldspar's multi-level associative retrieval. Faceted search allows the user to find an item by incrementally specifying its characteristics *internal* to the item itself. Multi-level associative retrieval, on the other hand, often requires both *internal* and *external* characteristics. For example, in the query "find the *file* that was received from *Bob*, authored by Sue, and *modified yesterday*", *modified yesterday* is an internal characteristic of the *file*, while *Bob* is an external one. These two retrieval strategies require different algorithms.

## INTRODUCING FELDSPAR

Feldspar is the first tool that fully supports multi-level associative retrieval of personal information. We first show how Feldspar works in action through an extended example, which touches upon all the major features of Feldspar. After that, we will describe the features in greater depth.

### Extended Example Illustrating Interactions in Feldspar

We will use the example from Figure 2: *find the webpage mentioned in the email from the person who I met in May*.

Bringing up Feldspar, the user selects *Webpage* in the first column to indicate that the desired item is a webpage. The result is shown in Figure 3a. Note that in the figure, Feldspar is already showing a set of possible answers. Here, they are the most recent web pages viewed by the user. The desired target web page is not displayed yet, so the user continues refining the query.

Next, the user wants to add *email* as the first association, because it is what the *webpage* is immediately related to. To add *email*, the user first clicks on the *related to* button to bring up the *refine panel (*Figure 3b). This panel provides the user with several ways to construct the association. The panel has rows for each data type. At the leftmost position of each row is the clickable name of the data type of that row. In the middle of each row are the top three suggested values of that type that Feldspar thinks are most relevant to the query, as determined by the Google Desktop's sort order. The *Date* data type, as an exception, has six suggested values, instead of three. We plan to experiment with different machine learning techniques, such as those that improve suggestions by examining users' past selections, to generate more meaningful sort orders, and therefore more relevant suggestions.

Back to the example, since the user does not remember which *email* contains the *web link*, the user would just click on the word *Email* at the left end of the row in the refine panel, and that creates an association column with *Email* selected.

Using a similar process, the user adds the third column for *Person*, and the fourth column for *Event*. The final thing the user remembers is that the event is in May, which is a specific time range that the user can indicate in Feldspar. Clicking on the *related to* button brings up the refine panel for dates. The month of *May* is not one of the top choices in the *Date* row (see Figure 3b), so the user clicks the *more* button at the right end of the column to bring up the *date picker* for more options (see Figure 2, at (8)). Here the user selects *May*. This completes the whole query. The web address is shown in the Query Results Area (Figure 2, at (3)), and the user can double click to open it in the web browser.

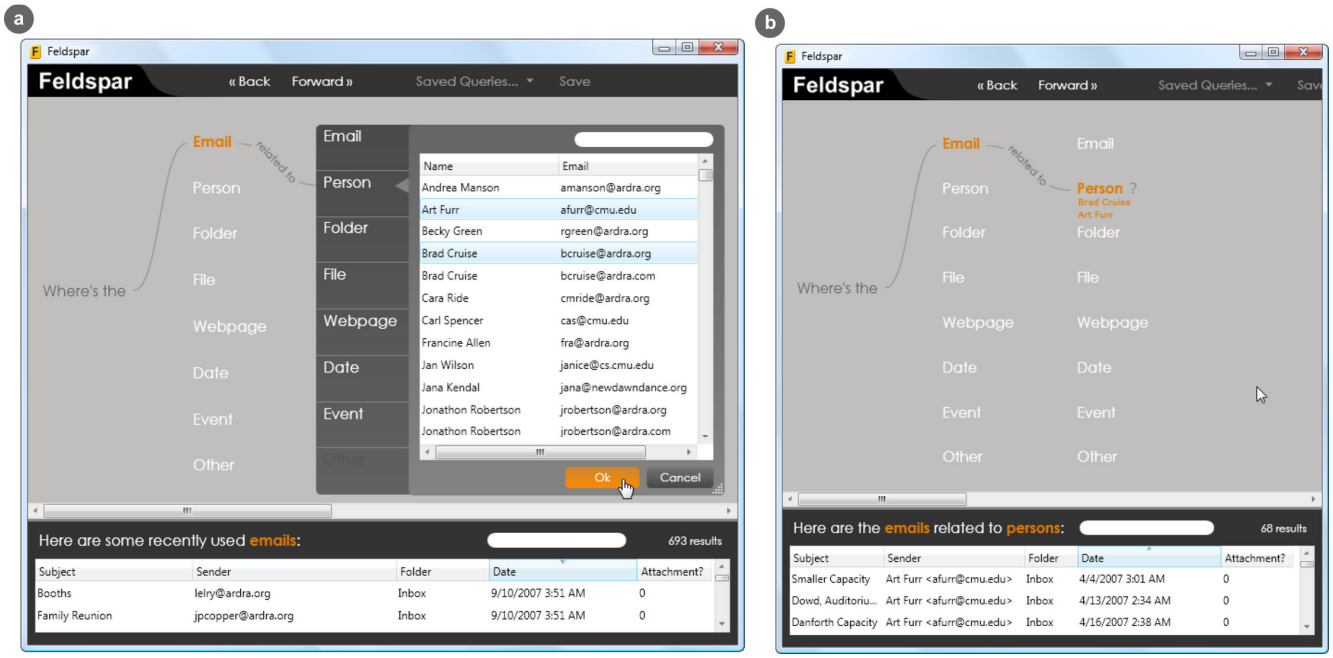


**Figure 4.** (a) The suggestion panel for Person, with two people selected. (b) Two people selected as constants for the second column.

## USER INTERFACE

The Feldspar user interface is composed of three main areas (see Figure 2). The Navigation Bar at the top (1) contains the Back and Forward buttons for moving to the previous and next screen. Underneath is the Query Area (2) for constructing the query visually and interactively. Finally, below the Query Area is the Results Area (3).

The Query Area is primary space where the user interacts with Feldspar. The user can incrementally construct a query and immediately see the updated query results at the Query Results Area. The query is presented as a question that begins with "Where's the …" and the user selects the desired type of item by clicking on the corresponding data type in the first column.

The user can mouse over a column to have the *frame* of the column to show up, as shown in Figure 2(5), and on top of the *frame* are the *header* and the *close button* for the column. Clicking the close button removes the column. Clicking and dragging the header can move a column around and exchange its order with other columns.

Recall in our extended example that the user brings up the *Date picker* (8). Other data types besides *Date* also have their own suggestion panels that can be brought up by clicking on their corresponding *more buttons* in the *refine panel*. Figure 4a shows the *suggestion panel* for *Person* with two people selected. After clicking "Ok", Feldspar shows the selected values (Figure 4b). If the user wants to change the selected values, clicking on the *Person* label or the constants in Figure 4b will return to the *refine panel* (Figure 4a) to enable editing. To cancel the selection of values, the user can double click on the selected type. The double-clicked column would be removed and the refine panel would re-appear so that the user can select another type for the column again. All of the supported types except Date currently use a list view like Figure 4a as their refine panel.

## IMPLEMENTATION

Feldspar is developed with Microsoft Visual Studio 2008 beta 2 and Microsoft Expression Blend 2 (for its UI), and is written in C#, utilizing features from Microsoft .NET 3.5 beta 2 programming framework.

Feldspar is a heavily data-driven application; it maintains a graph data structure, called the *association graph*, which stores the association information among items. The user interacts with this data structure though Feldspar's user interface, which serves as both a query construction tool and query results presentation tool.

### Obtaining Items via the Google Desktop Database

Before generating any associations among information items, we must first obtain the items themselves. We describe how we do this in the following sections.

#### *The Seven Item Types*

We are using Google Desktop to create the database for indexing and keeping track of the information items used by Feldspar. We can query the database by using the Google Desktop Search API.

Potentially, there are many types of information on a person's computer, so we must choose which to focus on for Feldspar. We did not start from scratch; instead, we first looked at the information that Google Desktop indexes, which includes a person's emails, files (including image files), visited web pages, calendar events from the Outlook calendar, media files, and more. We also examined its

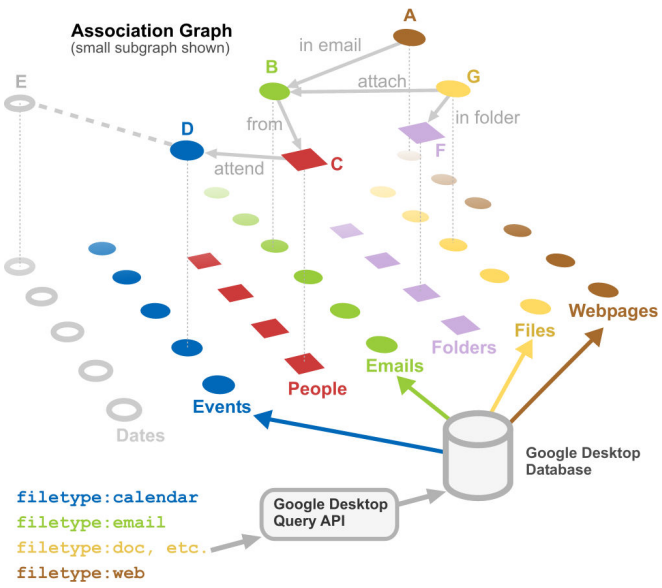


**Figure 5.** *Bottom*: Indexed items can be extracted from the Google Desktop database via the Google Desktop Query API. Items represented as solid circles, such as Emails, can be directly retrieved from the database. Square items are extracted from the circle items. Date items are dynamically generated during run time and are not stored in the association graph *Top*: The association graph links related items together. Its edges are labeled and directional.

Timeline feature that allows people to search by date. Based on what Google Desktop supports and what has been provided by prior systems, we decided to initially support a total of seven data types, which we thought were the most common things that people need to find. They are *Email*, *Person*, *File*, *Folder*, *Webpage*, *Event*, and *Date*. Eventually, we could add more types, but these serve quite well as a proof of the concept.

*Retrieving Emails, Events, Web Pages, and Files*

Email, Event, Webpage and File items (shown as solid circles in Figure 5) can be directly retrieved from the Google Desktop database through its Query API. For example, to retrieve *Email* items, we pass in the query string `filetype:email` to the API, and Google Desktop returns all of its indexed emails (shown as green circles) as *result objects*, the same format used by all information items returned by Google Desktop. Many relevant pieces of information about the email are included in the result object, such as the sender's name and email address, the email subject, receiving date, etc.

Using a similar process, we retrieve objects of calendar events (shown as blue circles), visited web pages (brown circles), and files (yellow circles). For files, we queried for all the popular file types.

*Further Extraction for Person Items*

After retrieving *Email* and *Event* items, we extract *Person* items (shown as red squares) from the *to*, *from* and *cc* fields of emails and the *organizer* and *attendees* fields of events. We also get people from the index of Outlook's contacts.

*Identifying Files' Common Containing Folders*

All *File* items returned by Google Desktop contain the full file paths, so it is possible for us to identify the folders containing those files at run-time. However, in practice, the list of files is often very long, thus making it computationally expensive to dynamically generate the folder lists. As a solution, we created a *Folder* item (shown as yellow circles) for every unique folder identified from the full list of files, and record the folder locations in the *Folder* items in advance of run time.

*Generating Date Items at Run Time*

The *Date* data type items are not stored in Feldspar's association graph (or in the Google database); instead, they are dynamically generated from the date attributes of other data types, such as from the *date* field of an *Event*.

**The Association Graph: the Heart of Feldspar**

Information items and the associations among them naturally form a graph, where items are vertices and associations are edges. Thus, we store this information with a graph data structure – we call it the *association graph*.

The association graph is the central component in Feldspar. It keeps all the association information, yet it is relatively lightweight. It generally uses less than one twentieth of the hard disk space that the Google Desktop database requires. For example, on the first author's hard disk, Google Desktop's database is 1 GB, while Feldspar's is 50MB. We reduced the space requirement of the association graph by only including the minimal information needed for keeping track of associations. For example, we do not keep the indexed content or the identified keywords of an email in the *Email* object. We keep only the unique ID that Google Desktop has assigned to the item, and then at run time, we can retrieve that item's full information from the Google Desktop when needed.

*Constructing the Association Graph*

Currently, we construct the association graph using the *Graph Builder*, a software module that we developed separately from Feldspar.

The association graph is a directed graph, implemented using the QuickGraph 2.0 open source graph data structure. The graph is directed because certain associations, such as "*the email from the person*", are directional. The *Graph Builder* first gathers items of all types and stores them as vertices in the graph. Then it creates an edge between each pair of related items. For example, it looks at the *to* and *from* fields of an email to identify the email's *sender* and *recipients*, and then it retrieves the vertices in the graph that represent those people and builds edges between those vertices and the email's vertex. Likewise, it extracts people from the *organizer* and *attendees* fields of events, and associates the corresponding person items to the events. Follow-

ing this approach, we identify associations among all items to construct the complete association graph.

*Producing Query Results*

In this section, we describe how our algorithm generates the results for a given query. For easier discussion, we use the example query for finding "the *folders* that contain the *attachments* received through *email* from *Spence*".

As the first step, the algorithm transforms the query into the list of associations "folders – files – emails – Spence" that it recognizes. Then, the algorithm uses one *results generator* for every pair of association A—B (A items related to B items) in the query to generate intermediary query results. Specifically, every result generator takes in a list of B items and returns a list of A items that are related to the B items. In our example, we need the following three results generators: (1) folders—files, (2) files—emails, (3) emails—persons.

Our algorithm starts processing the query from the last pair of associations, the emails – persons (Spence) pair. The emails—persons generator locates the vertex corresponding to Spence in the association graph and examines all of its in- and out- edges that are connected to email vertices. This gives us the emails *to* or *from* Spence. All these emails are then aggregated into a list, which is the output of the generator.

The algorithm then passes this list of emails as the input into the second generator, the files—emails generator, which outputs a list of files that are related these emails. In our example, the files would be attachments on any of the emails.

Finally, the algorithm inputs the list of files to the last generator, the folders—files generator and obtains a list of folders that are related to the files. In our example, they are the folders that contain any of the files which have been stored from the email onto the hard disk. The folders are also the final query results displayed to the user.

This chaining mechanism is efficient in producing query results, and more importantly, it allows the addition of new data types and their querying without having to change the algorithm itself. We just need to implement the new results generator for the new data types, which will be very manageable since Google Desktop supports only a small number of data types. In our current system, there are a total of 7x7 = 49 possible generators that can be implemented. However, not all pairs of associations make sense. For example, folders cannot currently be related to events. Of the 49 pairs, we implemented 38.

Some of the pairs required some thought as to what they would mean. For connecting files to people, we can use the author property that the file system maintains, but we also include files sent by that person in an email (implicitly adding a "...related to email related to..." in between the person and file). We will be further experimenting with which associations people expect Feldspar to keep track of for the various combinations.

**Scalability and Optimization**

We have not evaluated the scalability of Feldspar. However, from the experience of the first author using Feldspar on three of his computers, and also from the responses of the participants of our user study (see the Evaluation section), we have observed that Feldspar has been very responsive during run time, even when a complex query is being constructed and the database is large.

However, our current implementation of the association graph is not optimal. We currently store the graph on the hard disk, and load it into memory every time Feldspar is launched. This incurs a start-up delay that varies from a few seconds to up to about 10 seconds depending on the graph size. However, once the graph has been loaded into memory, Feldspar runs smoothly. We plan to implement a graph database for storing the association graph, which would allow us to keep the graph on the hard drive and load the necessary information from the graph on demand.

## EVALUATION

Feldspar is designed for people to use when they remember something associated with what they are trying to find. To try to simulate this in a controlled laboratory experiment, we told subjects what they should pretend to remember, to see if that would be sufficient for finding the target information. This setup is for evaluating the usability of Feldspar's user interface. We plan to evaluate Feldspar's effectiveness over extended use in a future longitudinal study.

**Participants**

Eight participants volunteered for our study, by signing up at an experiment website managed by our university. Their ages ranged from 20 to 39, with an average age of 26. Two participants were female. All participants were screened for their familiarity using Google Desktop and Microsoft Outlook 2003 for reading and writing emails and scheduling calendar events. Each study lasted for about 75 minutes, and the participants were paid $15 for their time.

**The participant as the Computer Owner**

We told the participants that they would be finding or looking at information on a computer that we provided. We emphasized to the participants that those information would be unfamiliar to them, and therefore they should not try to make any inferences about the information. (e.g., they should not try to infer the content of a file by just looking at the file name.) We then asked the participants to pretend to be Blake Randal, the fictitious owner of the computer.

**Information on the Computer**

All participants used the same desktop computer that we provided, which was populated with fictitious emails, files, and calendar events, and visited web pages.

We imported into Outlook an email corpus, containing 711 emails, which was developed as part of the Radar project and is freely available for research purposes [17]. We modified the receiving dates of the emails, so that they spread randomly across April and September 2007.

For files, we downloaded the first 50 files returned by Google for the popular file types: pdf, doc, xls, and ppt, by doing wildcard searches (typing "filetype: doc" into Google returns search results of doc files.) We distributed the files randomly into some file folders, with various hierarchy depths, that we created inside *My Documents*. We attached some of the files to the emails in Outlook so they would be both email attachments and on the hard disk.

To create web page browsing history, we visited the top 30 most popular web sites in the US, as listed on Alexa. Some web addresses were mentioned in the content of emails that we downloaded. And we injected a few more in the emails for the experiment.

We created calendar events spreading across April through September 2007 in Outlook, and we assigned people appearing in the email corpus as the event attendees.

#### Experiment Design

The study used a within-subjects design, with two main conditions for completing tasks: the Feldspar condition, where participants used only Feldspar, and the Control condition, where participants used conventional desktop applications, including Outlook and its built-in browsing and querying mechanisms, Google Desktop, and the Windows Explorer.

The test consisted of 14 tasks that are divided into two blocks. Every participant completed the first block of tasks in one condition, and then moved on to complete the second block of tasks in the second condition. The order of the two conditions was counterbalanced.

We created matched sets of 7 tasks each, Task Set A and B, ranging from easy to very hard, and counterbalanced which set was used with which condition, to guard against any unintended differences in difficulty. We used two sets of tasks to ensure that the subjects did not remember the answers from one condition to the other. The two sets differed only in the specific values used for parameters such as the associated person name or date.

The dependent measures in the study were the task completion times and completion rates. The following three factors could affect the dependent measure: (1) Software – Feldspar or the control group software; (2) Task Set – the Task Set used with the Software; (3) Software Order – which Software was used first.

Participants were randomly assigned to one of the following four conditions, with an equal number of subjects in each condition.

(Feldspar + Task Set A) then (Control + Task Set B)
(Feldspar + Task Set B) then (Control + Task Set A)
(Control + Task Set A) then (Feldspar + Tasks Set B)
(Control + Task Set B) then (Feldspar + Tasks Set A)

#### Tasks

We designed the tasks based on an informal survey of situations in which people had problems finding information on their computers, but remembered things associated with the information. The tasks in Task Set A were:

1) Open the last email received on *July 27, 2007*
2) Open all the email attachments of type *.txt*
3) Find out who had email conversations with the person who sent out the file *file.doc*
4) Find out who attended the event in which *Cara* was present.
5) Find all the events that were attended by anyone who has sent you a file.
6) Open the file folders that contain email attachments from *Spence.*
7) Open the webpage mentioned in the email from the person you met in an event in *May*.

The specific values used in the tasks are shown above in italic. Task Set B used the same tasks in the same order, but with a different set of specific values.

Tasks 1, 3, and 4 were simpler, while tasks 2, 5, 6, and 7 were more difficult Our hypothesis was that for *simple* tasks, Feldspar would achieve performance *comparable* to that of the control software, and for *difficult* tasks, Feldspar would be *significantly faster* than the control in both efficiency and accuracy.

#### Procedure

Before the participants started with each task block, they were given instructions on the software that they would be using. For the Control software, we went over the browsing and querying mechanisms of Google Desktop and Outlook, with which the participants should have already been very familiar. For the Feldspar software, we gave an overview of the different parts of the Feldspar user interface, the interaction techniques used in constructing queries, and the types of information that Feldspar is able to find. For both conditions, the participants were allowed to ask questions during these overview periods.

Then, we moved on to the first block of tasks. We instructed the participants to work quickly and accurately for all tasks. They were told that they had four minutes to perform each task, and that they could not move on to the next task until either the current task was finished, or the four minutes had passed.

Before starting each task, participants were asked to face away from the computer screen. They were given the instruction for that task to read. They could ask clarifying questions about the instruction. When they were ready to

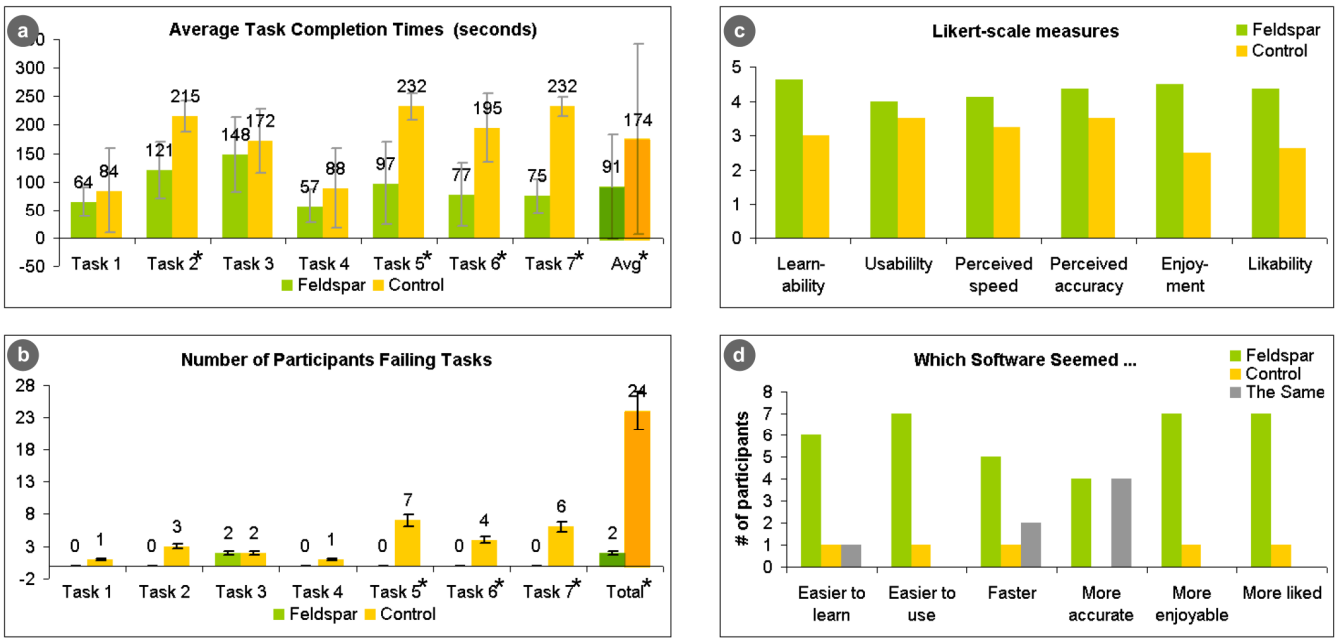


**Figure 6.** (a) Average task completion times. (b) Total number of failed tasks. (c) Likert-scale measures for both groups of software, with taller being better. (d) Participants comparing the two groups of software. For (a) and (b), error bars represent ±1 stdev and items with * are statistically significant.

begin, we asked them to turn around and press the start button of a timer located next to the computer mouse on the desk, and when they decided they were done with the task, the should press the stop button on the timer. Then the experimenter would check the results, and if incorrect, the participants were instructed to start the timer again and to keep trying. This self-timing approach was necessary so the participant's times would be stopped only when they *knew* that they had successfully completed the task, instead of when they were *told* so. This also provided more accurate task completion times. If the participants failed to finish a task within the allotted time, we stopped them, and recorded that as a failure.

After the participants finished the first block of tasks, they moved on to the next block. Finally, the participants filled out a questionnaire that asked for their subjective impressions about the software they used.

### Quantitative Results

The task completion time and complete rate data were analyzed using a mixed model analysis of variance with fixed effects for *software*, *software order*, and *task set*, and a random effect for *participants*. This analysis method is more appropriate for our within-participants study than a traditional ANOVA because individual error terms are synthesized for each participant [10].

#### *Task Completion Times*

We tested the task completion time data for the effects of all possible combinations of *software*, *software order*, and *tasks set* on *task time*, and we found significant effect only for *software*, indicating adequate counterbalancing for *software order* and matched difficulty for the two task sets.

Figure 6a shows the average task time for each task. Note that the maximum possible time is 240 seconds since we stopped subjects if they could not finish a task in 4 minutes, and we counted all failed attempts as taking 4 minutes. The main effect of software was significant on the completion times of task 2 ($F_{1,7}$=25.31, p<.0015), task 5 ($F_{1,7}$=30.16, p<.0009), task 6 ($F_{1,7}$=17.75, p<.0040), and task 7 ($F_{1,7}$=225.00, p<.0001). In other words, participants were significantly faster when they completed these four tasks with Feldspar. Incidentally, those four tasks were also the more difficult ones for the Control software. These tasks involve multi-level wildcard searches, which Feldspar greatly simplifies. For example, for task 7 – open the webpage mentioned in the email from the person you met in an event in May – participants using the control software would have to look at all the events in May, and then for each event, they would need to find all the attendees who have sent an email, and for each email, check for web links in the email contents. Overall, the average difference in time (91 vs. 174 seconds, or a factor of 1.91, almost 2 times slower) was also statistically significant ($F_{1,7}$=41.71, p<.0003).

#### *Task Completion Rates*

Using similar analysis, we tested the task completion rate data, and found significant effect only for *software*. Figure 6b shows, for each task, the number of participants who failed. Significantly more participants failed task 5 ($F_{1,7}$=49.0, p<.0002), task 6 ($F_{1,7}$=7.0, p<.0025), and task 7 ($F_{1,7}$=21.0, p<.0025) in the control condition. The total number of failed tasks was 24 for the Control condition, com-

paring to only 2 for the Feldspar condition; this difference is statistically significant ($F_{1,7}$=24.20, p<.0017), showing participants were dramatically more successful in finishing tasks when using Feldspar (a factor of 12).

Feldspar performed well across all tasks, in terms of both task completion times and completion rates. And importantly, Feldspar maintained almost constant performance time even for difficult tasks – with Feldspar, the times for the difficult tasks were not much different than the times for the easier tasks. In contrast, the Control condition suffered severely – participants either took much longer to complete those tasks, or failed them.

### Subjective Results

As measured by 5-point Likert scales filled out at the end of the study, participants felt that Feldspar was better than the Control software in all of the 6 aspects asked (see Figure 6c). This is a very encouraging result. More importantly, the participants enjoyed using Feldspar and found it easy to use. Furthermore, most participants perceived Feldspar to be easier to learn, easier to use, more enjoyable and better liked (Figure 6d). One participant commented "[Feldspar] is helpful to accomplish some complex set of activities. Helps a lot because you can relate data while searching. Extremely easy to use." Also, all participants felt that Feldspar was either more accurate than or as accurate as the Control software.

## DISCUSSION

The study results were positive, both quantitatively and qualitatively, confirming Feldspar's interface to be highly usable for the test tasks. We believe the most important factor that contributes to Feldspar's success is that it allows the user to easily take advantage of the *connections* (associations) between entities (pieces of information) when retrieving information. Although this may seem to be what some search algorithms, such as PageRank [2] have already been doing, there is an important difference – Feldspar provides users with a simple way to specify the *connections that they want to use*, while typical search programs attempt to choose associations automatically. For complicated tasks, like those from the user study, it is unlikely that search tools could easily guess what connections to use. Furthermore, search engines are not designed to handle the multi-level connections that Feldspar can express.

Another important factor is Feldspar's ability to chain together *non-specific* constraints (list of associations where only the association types are specified) to produce *specific* results (items that match the constraints). That is, Feldspar returns results even before a constant value is provided for the last column. Often, the user will find the result and stop before the query is even finished being formed. This is something that today's search tools cannot handle at all. The feature is very useful because although people often have difficulties remembering the exact details about the things they want to find, they usually have some *general* ideas about them.

The approach used in Feldspar actually focuses more on the *connections between entities*, and much less on the *entities themselves*. Similar approaches have been used in other domains, such as in social network analysis, detection of fraudulent transactions in online marketplaces, finding terrorist networks, and we expect to see even more examples in the future. We believe this is a natural trend, because as the number of information items increases, so do the number of connections between them. These connections often tell us many new things about the individual items, which may not be found if we just inspect the items in isolation.

However, we believe the associative approach will not be replacing but, rather, complementing the search and browsing approaches. For example, Feldspar currently does not look into the contents of emails. However, we could imagine incorporating Google Desktop's full-text search function into Feldspar such that we can even build queries that involve associating an item with another item that *contains* certain text.

## FUTURE WORK

Feldspar, in the current version, shows that associative information finding can work well, and it provides many features that people may find helpful. We have also designed a number of other features that are not yet implemented. We share these ideas here and hope they will stimulate discussions and help inspire even more design ideas.

We have used the general term "related to" to describe the association between items. In the future, we would want to allow users to change "related to" into a more specific association. For example, users would be able to select emails "from" or "to" people, or people who "attend" or "organize" events. In the user interface, we would provide a menu with the possible associations, which would pop up when a user clicks on the "related to" text or link. We note however, that the more general "related to" seems to work surprisingly well, and the specialization would only be needed when there are too many results.

Similarly, we would also want to allow people to specify whether they want to do an AND or OR across the multiple values that they select for a type. Currently, selecting email A and email B as values produces a query for items related to email A OR email B. We would also want to allow people to draw multiple "related to" links out of an item, to find items related to multiple other items of various types.

Currently, associations used in Feldspar are those that are easily detectable. In the future, we would like to support many more kinds of associations, some of which would require tapping more into the operating system to obtain. For example, we could associate two files together if we detect some data is copied from one file and pasted into the other. Furthermore, we could employ entity resolution algorithms [11] to resolve people having several email ad-

dresses or variations in the spelling of their name; mapping those email addresses and name variations to the same person in Feldspar internally.

Currently Feldspar obtains its data from the Google Desktop database. Alternatively, other database, such as the Windows Desktop Search or the Macintosh Spotlight database, could have been used, to make Feldspar more portable. We also want Feldspar to support gathering data from more sources, like from the Palm Desktop calendar and contact list. Another idea is to allow users to define additional sources of data associations. For example, to identify people *related to* a conference event, Feldspar could be given the list of authors or attendees.

Finally, we hope to release Feldspar for general use and collect feedback and ideas from the community about its usefulness and how to improve it.

## CONCLUSIONS

We have presented Feldspar, the first system that supports multi-level associative retrieval of desktop information. Specifically, Feldspar provides a novel interface that allows people to easily construct, edit and visualize a chain of associations as retrieval query. Indeed Feldspar is powerful in that it allows people to find things with non-specific requirements (such as using only data types). More importantly, the non-specific requirements can be chained together to produce specific results. Feldspar could be a useful addition to search and browsing, extending the ways people find and manage their personal information.

## ACKNOWLEDGMENTS

We thank Ken Mohnkern for creating a great demonstration video for showcasing Feldspar. This material is based in part upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010.

## REFERENCES

[1] Ahlberg, C., Williamson, C., and Shneiderman, B. Dynamic Queries for Information Exploration: An Implementation and Evaluation. In *Proc. CHI 1992*, ACM Press (1992), 619–626.

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

[3] Davies, G. and Thomson, D. *Memory in Context: Context in Memory*. Wiley, England, 1988.

[4] Dourish, P., Edwards, W. K., LaMarca, A., and Salisbury, M. Presto: An experimental architecture for fluid interactive document spaces. *ACM Transactions on Computer-Human Interaction (TOCHI) 6*, 2 (1999), 133–161.

[5] Freeman, E. and Gelernter D. Lifestreams: A storage model for personal data. *ACM SIGMOD Record 25*, 1 (1996), 80–86.

[6] Jones, W., Bruce, H., Foxley, A., Munat, C.F. The Universal Labeler: Plan the project and let our information follow. In *Proc. ASIST 2005*, Charlotte, NC.

[7] Kleek, M., Bernstein, M., Karger D., and Schraefel M. GUI— Phooey! : The Case for Text Input. In *Proc. UIST 2007*, ACM Press (2007), 193–202.

[8] Lamming, M. G. and Newman, W. M. Activity-based Information Retrieval: Technology in Support of Personal Memory. *IFIP Congress, Vol. 3* (1992), 68–81

[9] Lansdale, M. The psychology of personal information management. *Applied. Ergonomics 19*, 1 (1988), 55–66.

[10] Littell, R. C., Milliken, G. A., Stroup, W. W., and Wolfinger, R. D. *SAS System for Mixed Models*. Cary, North Carolina: SAS Institute, Inc, 1996.

[11] McCarthy, J. E. and Lehnert, W. G. Using Decision Trees for Coreference Resolution. *International Joint Conference one Artificial Intelligence 14*, 2 (1995), 1050–1055

[12] Nardi, B. and Barreau, D. "Finding and reminding" revisited: appropriate metaphors for file organization at the desktop. *ACM SIGCHI Bulletin* 29, 1 (1997), 76–78.

[13] Rekimoto, J. Time-machine computing: a time-centric approach for the information environment. In Proc. *UIST 1999*. ACM Press (1999), 45–54.

[14] Rhodes, B.J. The Wearable Remembrance Agent: a system for augmented memory. In *Proc. ISWC 1997*, Cambridge, Mass., USA (1997), 123–128.

[15] Ringel, M., Cutrell, E., Dumais, S. T., and Horvitz, E. Milestones in Time: The Value of Landmarks in Retrieving Information from Personal Stores. In Proc. *INTERACT 2003*, 184–191.

[16] Rothrock, B., Myers, B.A., and Wang, S.H. Unified Associative Information Storage and Retrieval. *Ext. Abstracts CHI 2006*. ACM Press (2006), 1271–1276.

[17] Steinfeld, A., Bennett, S.R., Cunningham, K., Lahut, M., Quinones, P.-A., Wexler, D., Siewiorek, D., Hayes, J., Cohen, P., Fitzgerald, J., Hansson, O., Pool, M., and Drummond, M. Evaluation of an Integrated Multi-Task Machine Learning System with Humans in the Loop. In *Proc. NIST Performance Metrics for Intelligent Systems Workshop (PerMIS), NIST (2007)*.

[18] Teevan J., Alvarado C., Ackerman M., and Karger D. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *Proc. CHI 2004*. ACM Press (2004), 415–422.

[19] Tulving, E. and Thomson, D. Encoding specificity and retrieval processes in episodic memory. *Psychological Review 80* (1973), 352–373.

[20] Yee, K., Swearingen, K., Li, K., and Hearst, M. Faceted metadata for image search and browsing. In *Proc. CHI 2003*. ACM Press (2003), 401–408.