Feldspar: A System for 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
We present Feldspar, the first system that allows people to find personal information on the computer by specifying chains of other information that it is associated with, emulating the information retrieval process of human associative memory. Feldspar's contributions include (1) a user interface for constructing retrieval queries that consist of *multiple* levels of associations, such as "find the *folder* containing the *email attachment* from the *person* I met at an *event*"; 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 is superior to conventional browsing and searching for these kinds of retrieval tasks. We have reported Feldspar's implementation and evaluation in our long paper at CHI 2008. Here, our discussion focuses on the design and implementation ideas that we have tried, or are currently investigating. We hope this will stimulate further discussions and help inspire more design ideas.

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
People can often recount chains of associations when trying to remember things [4, 18], like "I remember receiving an *email* from a *person* who I met at an *event* that happened in *May*", although they may not remember details about the things themselves. Current search and browsing tools are unlikely to help the user find the desired email in this example, because intrinsic details of the email, including its location and the text within it, are unknown. However, a system that supports *associative* retrieval – where the user finds things by specifying *other* things that go with them – may have worked. Therefore, we created the Feldspar system (see Figure 1), the first tool that supports multi-step associative retrieval of personal information on the computer, which emulates how human associative memory works. Feldspar is more thoroughly described in a companion article [3].

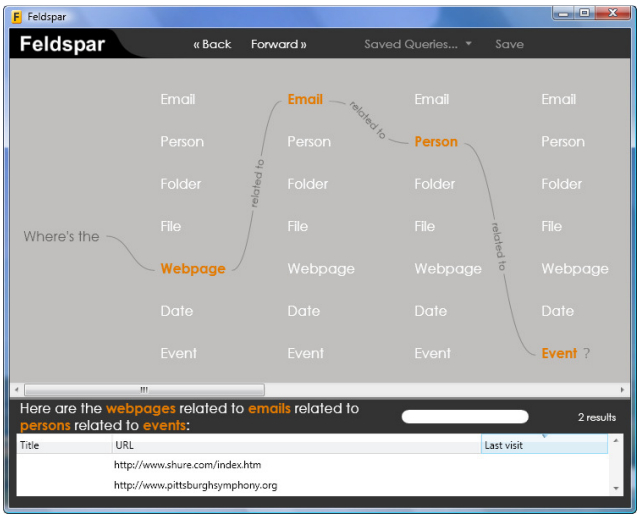

Figure 1. The Feldspar user interface showing a user requesting the *webpage* mentioned in an *email* from a *person* related to an *event*.

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 highly usable in a user test.
- Algorithms for collecting the association information and for providing answers to associative queries in real time.

INTRODUCING FELDSPAR
Feldspar stands for **F**inding **E**lements by **L**everaging **D**iverse **S**ources of **P**ertinent **A**ssociative **R**ecollection. Its user interface is composed of three main areas (see Figure 3). The Navigation Bar at the top (1) contains the Back and Forward buttons for moving to the previous and next

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

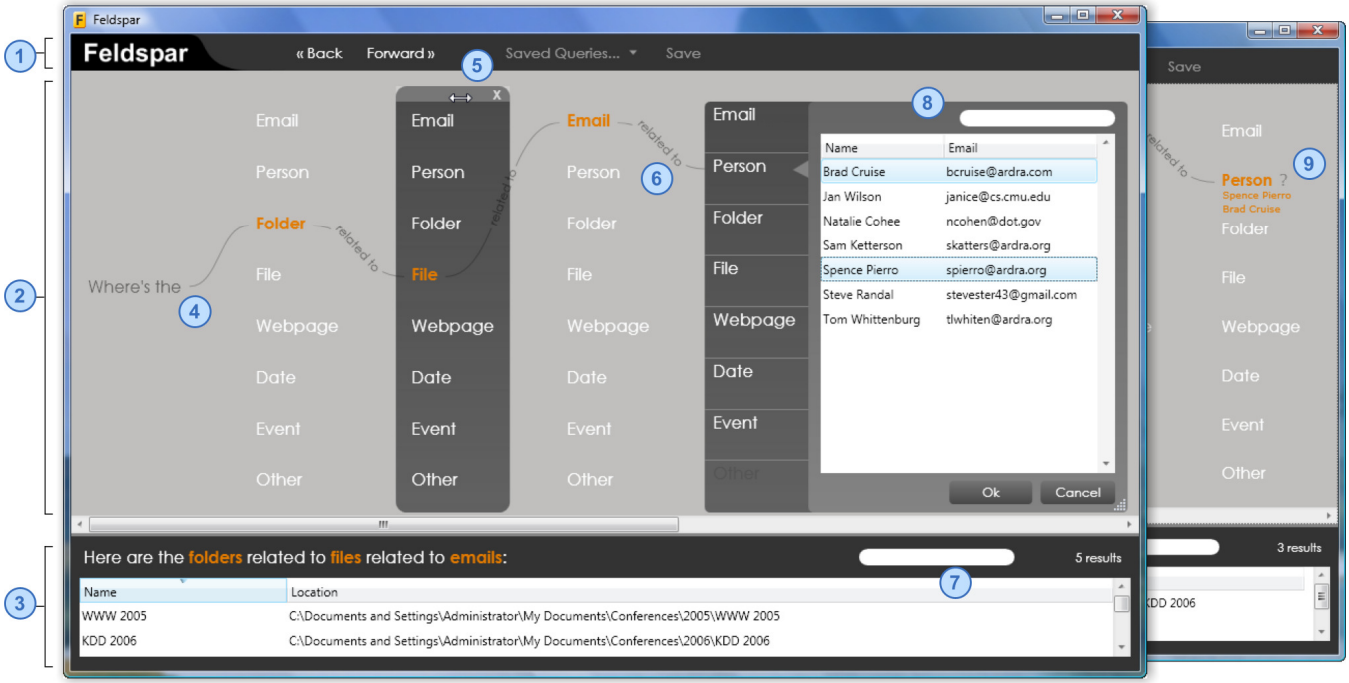

Figure 2. The Feldspar user interface. 1: The Navigation Bar. 2: The Query Area for constructing queries. 3: The Results Area with the query represented as a sentence at the top. 4: The main query area. 5: The user can click to edit the type of an association and swap its order with other associations. 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 suggested list of people for the user to pick. Multiple values can be selected by shift-clicking on them. At any time, the user can edit the query by selecting different values and the results update immediately.

screen. Underneath is the Query Area (2) for constructing the query. Finally, below the Query Area is the Results Area (3).

The Query Area is the primary space where the user interacts with Feldspar. The user can incrementally construct a query by adding one association after another and immediately see the updated query results for that intermediary query at the Query Results Area, so that 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].

The query is presented as a question that begins with "Where's the …" (Figure 3, at (4)) 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 an association column to have the *frame* of the column to show up, as shown in Figure 3 (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.

Extended Example Illustrating Interactions in Feldspar

We show how Feldspar works in action through an extended example from Figure 3: *find the folder containing the email attachments received through email from Brad or Spence*. This example will also introduce other parts of the interface that we have not yet mentioned.

Bringing up Feldspar, the user selects *Folder* in the first column to indicate that the desired item is a folder. 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 folders accessed by the user. The desired folder is not displayed yet, so the user continues refining the query, by clicking the *related to* button which brings up the *refine panel (*Figure 3b). This panel provides the user with several ways to define the next 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. Each row has a *more* button at its right end. Clicking on it will show the full list of suggested values (Figure 2, at (8)). The *Date* type is an exception; it has six suggested values on the refine panel, and a *date picker* panel (Figure 3c) is shown instead of a suggestion list when the *more* button is clicked.

Back to the example, since the user does not remember which files are contained in the target folders, the user would just click on the word *File* at the left end of the *File* row, and that creates an association column with *File* selected. Using a similar process, the user adds the third column for *Email*. The final thing the user remembers is

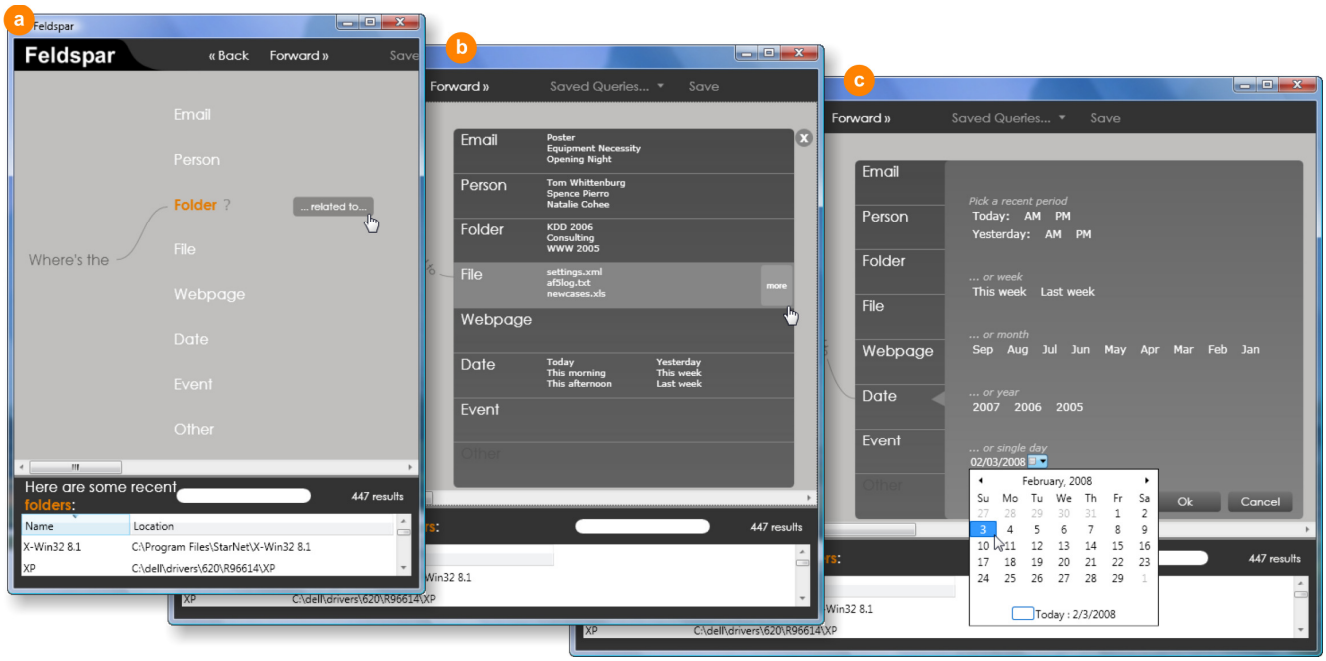

Figure 3. (a) Feldspar after the user has selected *Folder*. (b) The *refine panel*, which appears when the user clicks the *related to* button. (c) The *date picker panel*, which allows the user to pick a data range, such as the month of May, or a specific date using a calendar dropdown.

that the email is from *Brad* or *Spence*, which are values for the *Person* type. To specify them, the user clicks the *related to* button to bring up the *refine panel*, and then clicks the *more* button at the right end of the *Person* row to bring up the list of people suggested by Feldspar. The user sees *Brad* and *Spence* and selects them both by shift clicking on them (see Figure 2, at (8)). This completes the whole query. The folder is shown in the Query Results Area (Figure 2, at (3)), and the user can double click to open it.

IMPLEMENTATION*

Feldspar is written in C#. It uses the database maintained by Google Desktop to keep track of information items on the computer (e.g., emails, files, etc.). Feldspar accesses these items through the Google Desktop Search API.

Storing Association Information

Feldspar analyzes the information items and identifies the associations among them, and stores this information with a graph data structure – we call it the *association graph* – where items are vertices and associations are edges.

The association graph is a directed graph, because certain associations, such as "*emails from people*", are directional. To create this graph, Feldspar 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 builds an edge, labeled with a *to* attribute, from an email item to a person item if the email is sent to that person. Following this approach, we identify associations among all items to construct the complete association graph.

Retrieving Information Items from Google Database

Based on what Google Desktop supports and what has been provided by prior systems, we decided to initially support seven data types that we thought were the most common things that people want to find. They are *Email*, *Person*, *File*, *Folder*, *Webpage*, *Event*, and *Date*.

Producing Query Results

Feldspar uses an iterative algorithm to generate the results for a given query. For easier discussion, we use the example query for finding "the *attachments* received through the *email* from *Spence*". Using Feldspar's interface, the query is represented as the list of associations "files – emails – persons (Spence)". 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. Each 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 two generators: (1) files—emails, (2) emails—persons.

The algorithm can handle queries of arbitrary length consisting of any types of associations, provided that the results generators for those associations are implemented. As Google Desktop supports only a small number of data types, and that not all pairs of associations make sense, the growth in the number of generators will be very manageable. In our current system, there are 7x7 = 49 possible generators, and we implemented 38 of them for now.

* Please refer to our long paper at CHI 2008 [3] for more details on Feldspar's implementation and evaluation.

EVALUATION

We evaluated the *usability* of Feldspar user interface through a small laboratory study. Eight participants volunteered. They were familiar with using Google Desktop and Microsoft Outlook 2003 for reading and writing emails and scheduling calendar events.

We populated a desktop computer with fictitious emails, files (including email attachments), calendar events, and visited web pages. We asked the participants to pretend to be Blake Randal, the fictitious owner of the computer, and to use Feldspar to find information on the computer.

The study compared two main ways for completing the 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. We told the participants to work quickly and accurately for all tasks. They had four minutes to perform each task. If the participants failed to finish a task within the allotted time, we stopped them, and recorded that as a failure.

Task completion times were found to be affected only by *software* (Feldspar versus Control). Participants were significantly faster when using Feldspar to complete the more difficult tasks. These tasks involve multi-level wildcard searches, which Feldspar greatly simplifies. Overall, the difference in average task completion time between the two conditions was 91 vs. 174 seconds, with Feldspar being almost twice as fast. This was statistically significant ($F_{1,7}$=41.71, p<.0003).

Using similar analysis, we found that task completion rates were, again, only affected by *software*. Overall, participants were dramatically more successful in finishing tasks when using Feldspar, with only 2 fails, compared to the 24 for the Control condition.

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. They 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.

PREVIOUS USER INTERFACES

Feldspar's user interface is highly usable for the test tasks. Before coming up with this interface, however, we experimented with several other versions of the interface. Here we share what we learned from that experience.

It seemed clear to us that to successfully perform associative retrieval, we needed to place strong emphasis on the relationships among information items, for designing both the user interface and the retrieval algorithm. We also knew that we should use a graph data structure to store the information items on the computer. However, coming up with a good user interface was a challenging task.

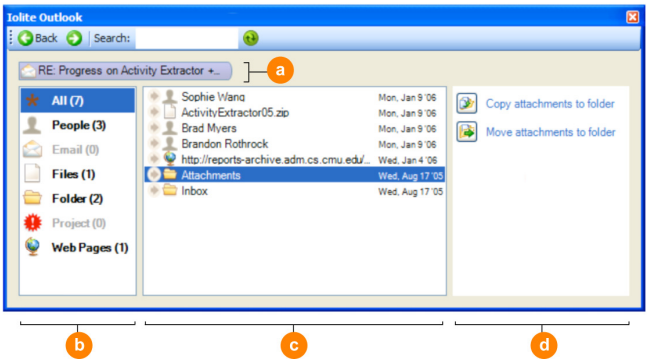

Figure 4. The Iolite user interface. (a) *Breadcrumb* showing the sequence of associated items the user has navigated. (b) Category filters for the item list shown to the right. (c) List of items immediately associated with the latest item in the breadcrumb. (d) Actions that can be performed with the selected item.

In the beginning, we tried to view the problem of associative retrieval as a graph navigation problem, where the user started with a specific information item, and navigated from it to the desired item through exploring the network of associations. Based on this methodology, we created the Iolite user interface.

The Iolite User Interface

We created our earlier Iolite [16] system with strong emphasis on designing some of the underlying algorithm for automatically discovering associations among information items. Iolite continuously monitored activities on the computer and tried to infer context through machine learning techniques. We put little emphasis on the user interface (see Figure 4), however.

One major limitation of the Iolite interface was that it could only work with *specific* items (general types, like *Email*, were not supported), making it difficult to simultaneously locate all the information items on the computer that shared similar sequences of associations. Another limitation was that, to locate a desired item, the user must navigate from *a specific item* (such as an email in the inbox), and *expand outwards*, by adding more associations, putting the *target* item as the *last* association. This approach could be limiting at times, because (1) locating a specific item to start with could be a difficult task by itself; and (2) over-specifying could easily occur during the intermediary steps, preventing the targets from being found. In addition, it was visually unclear that the list of items shown (see Figure 4c) was associated with the latest item in the breadcrumb chain.

To try and overcome this last shortcoming, we created a interactive prototype of an alternative design (see Figure 5. The prototype placed the currently selected item at the center, surrounded by its associated items, grouped by categories. Edges were used to connect the categories to the selected item to indicate their association. When the user

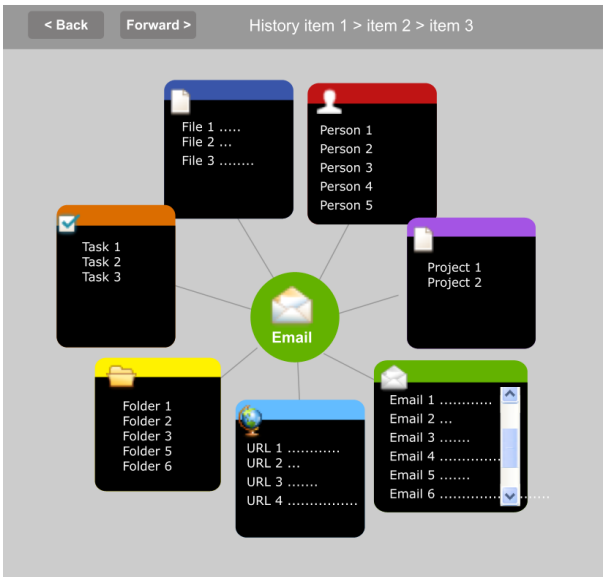

Figure 5. Another version of the Iolite interface. Currently selected item is shown at the center, surrounded by its associated items, grouped by categories

picked the next associated item from one of those lists, that item would animate and move to the center, replacing the previous item.

Although this design could intuitively help express the flow of association information, we did not adopt the design, because we found the animations introduced to the interface was too overwhelming. This design also demanded large amount of screen real estate, while only showing a few items for each category. Besides, each category "patch" could only be assigned a short width, limiting the number of attributes that we could show for each item.

Based on problems that we identified from these interfaces, we created the improved Feldspar interface shown at the beginning of this paper, which has overcome those shortcomings.

DISCUSSION

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 – with Feldspar, users can 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 *general types* as the query to produce *specific* results. That is, Feldspar can return useful 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 – with Feldspar, they can turn these general recollections into a query to find what they are looking for.

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 replace but, rather, complement 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. For example, we can find all the people who have received email attachments that contain the phrase "year-end report".

Performing long-term evaluation for personal information management (PIM) tools can be very challenging, because the tools need to be sufficiently robust to sustain long-term usage in real-world conditions. Keeping this in mind, we planned to make Feldspar as stable as possible, even before we started implementing it. We realized a major source of instabilities comes from having to process a large amount of raw information on the user's computer, to keep track of the processed information, and to continuously monitor the computer for new information for processing. We found that we could eliminate these potential pitfalls by letting existing information indexing systems do the work for us (Google Desktop). The system indexes the information on the computer, and maintains databases to keep track of it.

Popular indexing systems include the ones from Google Desktop, Windows Search, and Spotlight (for Mac OS X). We discuss the benefits of using an indexing system through the example of Feldspar:

(1) The raw information on the user's computer remains intact at all times. This could help relieve users' concern about potential data loss caused by Feldspar. Feldspar can access the information indirectly through the Google Desktop API.

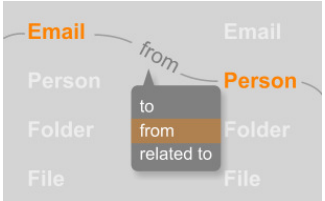

Figure 6. A popup menu for changing the type of association between *Email* and *Person* items.

(2) Indexing systems are available on all popular operating systems. We could make Feldspar more portable by programming it to support different databases.
(3) Users can use Feldspar without having to change how they store or organize their information, because Feldspar can find all the information on the users' computer through the indexing system.
(4) The indexing system can notify Feldspar when new information is indexed, through the indexer's event framework. This will allow Feldspar to work with the latest information available.

We encourage fellow researchers to leverage indexing systems when implementing their PIM systems.

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.

Specializing Associations

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 (seeFigure 6). 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.

Streamlined Interface

Feldspar was designed to maximize usability. Every column in the interface contains the most common data types that

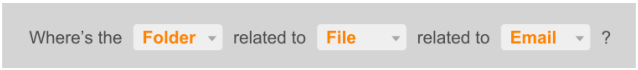

Figure 7. A streamlined version of the Feldspar interface for finding folders that contain email attachments.

Feldspar support, allowing the user to quickly scan through them, and pick a type with one click. As an alternative design, we could replace each column with a drop-down menu (see Figure 7). This will significantly reduce the vertical dimension of the interface, but this will likely reduce the discoverability of available data types and increase the time for making selections. When screen real estate is limited, however, this streamlined version is preferable.

More Associations

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. We could also associate a webpage with a file that we have just downloaded from that page.

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. We also want Feldspar to support gathering data from more sources, like from the Palm Desktop calendar and contact list.

Using Feldspar to Return to Previous Working Context

Associating together files that shared the same copied-and-pasted text, and associating webpages with downloaded files, as mentioned in the previous section, are just some of the many associations that could help bring users back to their previous working context. Imagine a user using Feldspar to reload all the files that were edited together, and to re-find the webpage that a file was downloaded from.

Extending Feldspar to Support Subgraph Matching for General Graphs

Feldspar can be viewed as a *graph pattern matching tool*. It allows the user to *specify* and *construct* a subgraph pattern (as a chain of associations), and tries to find all the matching subgraphs within the large graph of items.

This implies the Feldspar interface might also be suitable for building queries, in the form of subgraphs, for other type of problems. For example, Feldspar can be easily modified to allow the construction of queries for loops, which could represent interesting patterns in certain domains (money-laundering schemes contain loops, for instance).

RELATED WORK

Many research and commercial systems have been created to support various forms of search and retrieval. Popular desktop search programs such as Google Desktop, Microsoft Windows Desktop Search, and Spotlight for Apple's OS X allow the user find information items by keywords. Many systems support time-based retrieval, such as TimeScape [13] and the Lifestreams system [6]. Ringel, et al. [15] proposed a timeline-based visualization of search results of personal content, highlighting major events to help the user retrieve information more easily. Nardi and Barreau [12] suggested the need for better location-based document management systems, based on their observation that people often placed together similar documents. Jones, et al. [7] proposed a project-centric approach for organizing and retrieving electronic documents. Lamming and Newman [9] 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. Dourish, et al. [5] proposed an attributed-based method, similar to tagging, to help people categorize documents. Some more recent systems include the commercial tool Xobni (www.xobni.com) and the research system Jourknow [8]. All of the above systems only support one level of association, not the multiple levels that people often remember.

Although faceted search [19] 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.

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. It could be a useful addition to search and browsing, extending the ways people find and manage their personal information. We discussed many design and implementation ideas that we have tried, or are currently investigating, and we hoped this will stimulate further discussions and help inspire more design ideas.

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. Chau, D. H., Myers, B., and Faulring, A. What to Do When Search Fails: Finding Information by Association. In *Proc. CHI 2008*, Florence, Italy.
4. Davies, G. and Thomson, D. *Memory in Context: Context in Memory*. Wiley, England, 1988.
5. 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.
6. Freeman, E. and Gelernter D. Lifestreams: A storage model for personal data. *ACM SIGMOD Record 25*, 1 (1996), 80–86.
7. 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.
8. 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.
9. Lamming, M. G. and Newman, W. M. Activity-based Information Retrieval: Technology in Support of Personal Memory. *IFIP Congress, Vol. 3* (1992), 68–81
10. Lansdale, M. The psychology of personal information management. *Applied. Ergonomics 19*, 1 (1988), 55–66.
11. 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.
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. Tulving, E. and Thomson, D. Encoding specificity and retrieval processes in episodic memory. *Psychological Review 80* (1973), 352–373.

19. 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.