FacetMap: A Scalable Search and Browse Visualization

Greg Smith, Mary Czerwinski, Brian Meyers, Daniel Robbins, George Robertson, Desney S. Tan

Abstract— The dominant paradigm for searching and browsing large data stores is text-based: presenting a scrollable list of search results in response to textual search term input. While this works well for the Web, there is opportunity for improvement in the domain of personal information stores, which tend to have more heterogeneous data and richer metadata. In this paper, we introduce FacetMap, an interactive, query-driven visualization, generalizable to a wide range of metadata-rich data stores. FacetMap uses a visual metaphor for both input (selection of metadata facets as filters) and output. Results of a user study provide insight into tradeoffs between FacetMap's graphical approach and the traditional text-oriented approach.

Index Terms— Graphical visualization, interactive information retrieval, faceted metadata

1 Introduction

Searching electronic data collections has become increasingly widespread and important in the personal and professional lives of computer users. The Web, with its constantly changing and expanding corpus of documents, is the most obvious driver, and dozens of corporate players (including Google, Microsoft, and Yahoo) are investing heavily in research and technology aimed at making Web search easier, faster, and more rewarding. This investment in search is also being increasingly applied to the individual computer desktop, with many of the same corporate players producing versions of their tools for local data stores.

The personal computer is becoming more and more a personal data store, and while it may never rival the Web in the raw number of stored items, it is already long past the point where a simple folder hierarchy for documents is sufficient for a user to organize, search and keep track of their electronic artifacts. Recent commercial efforts, including desktop search from Yahoo (desktop.yahoo.com), Google (desktop.google.com), and Microsoft (desktop.msn.com), as well as native operating system support for search in Apple's OSX Spotlight and Windows Vista, explicitly address this by trying to unify all the information passing through a single machine – web pages, email, photos, files, etc. – into a single search index. But the research community has been considering the issue for much longer. Starting with Vannevar Bush's MEMEX vision in 1945 [6], many researchers have been attempting to design systems that act as "memory extenders" [11][12][14][17][19][20][22][23]. However, only a subset of these systems was designed to provide search and browse capabilities over one's entire personal store, and only one of these systems, Ringel et al's personal memory landmark system [23], was evaluated and shown to be effective with target end users. Additionally, many of these systems are heavily textual. Given the ongoing explosion of recorded content enabled by automated data collection mechanisms, a text-based interface may become increasingly overwhelming to users, especially novices. Hence, one of our primary goals was to explore designs for a personal information retrieval system that not only allows users to take advantage of their visual spatial abilities, but is also pleasing to use.

Currently, the dominant model for performing search remains keyword text entry followed by the presentation of a textual list of results in relevance order. This model has particularly suited the Web scenario because of its efficient use of network bandwidth and its uniform presentation on many different platforms and form factors. However persistent this model may turn out to be on the Web, a personal data store has several differentiating characteristics that make the predominance of the pure iterative keyword text search less certain for the future.

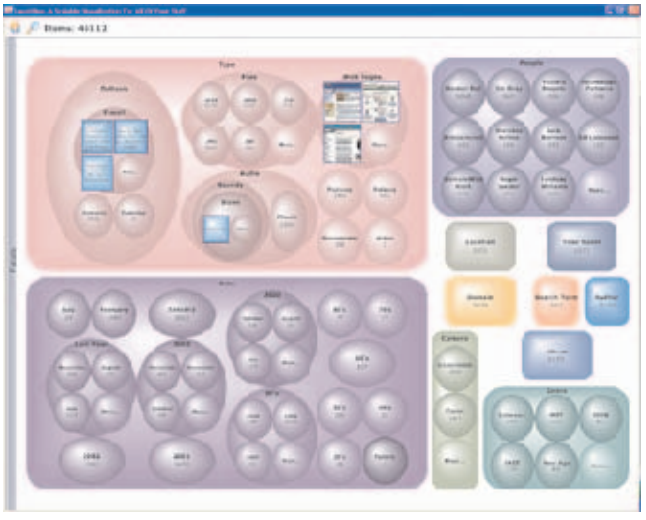

Fig. 1. Screenshot of the FacetMap interface.

A Web search favors precision over recall, since a typical search target on the Web exists alongside a million more "good enough" matches. But the personal data store is a more intimate domain – it may represent an individual's lifetime of documents, communications, and digitized experiences, which may not necessarily lend themselves to being searched with keywords or summarized in uniform list boxes. These search atoms also come with a much richer, more personally relevant set of metadata – widespread attributes like times, places, and types, as well as many more subset-specific attributes like the "camera type" attribute for pictures. While the Web favors specific searches for approximate targets, the personal store more often needs approximate searches for specific targets. The text-search/text-results paradigm does little to exploit the personal data store owner's familiarity with the corpus, to facilitate discovery of serendipitous knowledge about the makeup of the corpus during the search process, or to support browsing for the purpose of reminiscing or sharing. Finally, the limited interface requirements imposed by the simple keyword/list box paradigm are increasingly unnecessary handicaps on the personal store, leaving the processing power, graphics capabilities, and screen real-estate of a such a store severely underutilized.

- *Greg Smith is with Microsoft Research, E-Mail: gregsmi@microsoft.com.*
- *Mary Czerwinski is with Microsoft Research, E-Mail: marycz@microsoft.com.*
- *Brian Meyers is with Microsoft Research, E-Mail: brianme@microsoft.com.*
- *Daniel Robbins is with Microsoft Research, E-Mail: dcr@microsoft.com.*
- *George Robertson is with Microsoft Research, E-Mail: ggr@microsoft.com.*
- *Desney S. Tan is with Microsoft Research, E-Mail: desney@microsoft.com.*

Manuscript received 31 March 2006; accepted 1 August 2006; posted online 6 November 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

We created FacetMap to exploit these differences and to begin exploring a different approach to searching and browsing. FacetMap (Figure 1) is a visualization for interacting with large, metadata-rich databases by allowing the user to easily assemble complex queries through iterative, direct interaction with the graphical output of the system.

In the following sections, we review related work and discuss the principles and tradeoffs involved in our interface design and the underlying data infrastructure. We also present an initial user evaluation of the system, in which we not only benchmark FacetMap against a domain-optimized list-based query system, but also examine the success of our design with regard to users' mental models of faceted metadata search.

2 Related Work

2.1 Information Retrieval

Most of the recent progress in searching large data stores like the Web has come from a combination of improved precision in the underlying information retrieval task of keyword search (surfaced via "ranking" of text-based results) with more advanced technological implementations (faster machines, and more efficient storage and indexing). As discussed in the introduction, this is not the only possible approach, nor is it one that applies equally well to all interesting corpora and user tasks.

A common alternative for making large datasets tractable for interactive exploration is through the use of a browseable hierarchy, also known as categorization. When data items are aggregated into a small number of categories at each level, the user can leverage the power of a log *n* descent to rapidly find a single item out of many, just by iteratively choosing from among the categorized choices. One oft-cited example of this approach on the Web is the category view offered on the Yahoo search portal [dir.yahoo.com]. There are some well-understood drawbacks with this approach: it can be laborious to impose a post-hoc categorization on a large corpus, it is often impossible to agree upon a single 'best' categorization, and as the corpus evolves it may be difficult to keep a balanced tree – essential for maintaining an effective narrowing factor during descent. Indeed, Yahoo's category search has effectively been deprecated (by relegating it to a Yahoo sub-domain) in favor of investment in traditional text keyword search.

There have been several efforts [8][18][24][28] to make large databases more browseable through the use of automatic keyword clustering techniques, which reap some of the benefits of categorization while avoiding the laborious manual labeling and balancing requirements. While often useful for conveying understanding of a corpus as a whole, it appears that these automatic clustering techniques do this at the expense of the ability to search in a more targeted way, or to browse by criteria other than textual similarity [18]. Those that do offer additional integrated search capabilities [28] usually apply the clustering only *after* the initial keyword search is complete, creating a dual-mode interface. With FacetMap, we wanted to create something with equal facility in search and browse along any criteria the user has in mind, and to present a consistent interface throughout the search/browse experience.

Using "faceted metadata" as a refinement to the approach of categorization has been gaining traction in recent years. In this approach, the attributes (metadata) of the dataset items are grouped into multiple orthogonal categories called "facets." For example, a database of fine arts items might have a *Date* facet to group together the item creation dates, a *Location* facet to represent the creation locations, and a *Media* facet exposing attribute values like "painting," "photograph," or "sculpture." Presenting several facets simultaneously in a search and browse interface mitigates the shortcomings of any single categorization scheme and therefore supports a wider diversity of users and user tasks. It has been demonstrated in the Flamenco system that a significantly more efficient and enjoyable user experience (as compared to keyword search or pure categorization) can be achieved by integrating faceted metadata into a comprehensive dynamic query interface [27]. Another example of this approach is Phlat [7], which offers a faceted view of a personal store with particular emphasis on integrating user "tags" into the facet space. The faceted metadata approach has also gained commercial popularity in certain Web and Intranet scenarios, with several companies (e.g. Endeca [www.endeca.com], Inxight [www.inxight.com], and i411 [www.i411.com]) arising to capitalize on the opportunity.

FacetMap draws on several design innovations in these approaches, including the use of facets as a top-level organizational interface concept, deep integration of text and attribute searching, a data-driven dynamic query interface, and heavy use of query previewing along different facet axes. However, most faceted metadata-based interfaces have simply replaced the problem of laboriously categorizing the data items with the problem of laboriously categorizing the items' metadata into specific facets with just the right count and structure to fit the target interface, and the success of the interface is tightly tied to success in this process. Unlike interfaces which fix specific layouts for specific facets and values, FacetMap uses a fully dynamic allocation of screen space based entirely on the distributions of attributes among the remaining items in the result set. While the resulting spatial instability is a serious potential concern for usability, in our work we felt it was important to explore a technique that could be used across a wide range of data domains and search tasks by virtue of its independence from any particular metadata classification scheme.

2.2 Information Visualization

Visualization research on large data stores has progressed primarily in two ways: static, pre-computed visualizations of the large dataset itself, or more dynamic visualizations of a smaller, post-query subset, usually of a few hundred items resulting from an initial search on the original larger dataset.

Automatic clustering technology, mentioned earlier, exhibits some promise in visualizing a large dataset while still allowing some measure of interactivity. There have been several efforts to produce interactive 2D or 3D visualizations on top of this type of automated aggregation, such as [21]. In addition to the drawbacks already mentioned for clustering, the overall usability appears to suffer even further from the unpredictability and novelty of the graphical interaction. Zhang *et al.* [30] produced a dynamically-clustered, force-directed (embedded springs) visualization but they did not consider it usable enough to study it with end users. ThemeScape [13] is a visual representation of a large document collection spatially arranged by document similarity. Fabrikant [9] studied a similar system and demonstrated that users zooming into a spatial area understand that they are going deeper into a semantic hierarchy. With FacetMap we hoped to leverage this same innate understanding while using faceted hierarchical metadata in the spatial arrangement rather than document similarity

Aggregation of any sort necessarily hides detail by choosing a particular axis along which to collapse individual items into groups. Accordingly, some have proposed novel ways to scale visualizations to very large databases without this type of lossy compression in the visual presentation. Techniques have been developed for over-sampling individual pixels to allow the creation of accurate pre-attentive patterns even when each individual item theoretically has less than a single pixel of real estate [10][15]. But these techniques require each dataset item to be individually processed – meaning that either the result is static and pre-computed, or the entire dataset has to be represented in system memory in order to perform the dynamic transformations required by user interaction, which puts practical limits on the dataset size. One clever way to allow real-time interactivity in large, dense representations is by exploiting native Graphics Processing Unit (GPU) operations to perform queries [10]. However, offloading more of the visualization to the GPU, thereby decreasing the per-item information storage of the system, makes the

visualization less dynamic – e.g., remapping the color to a different facet of the dataset items would require a full reprocessing of the dataset.

In the case of smaller search result subsets of larger datasets, visualizations are able to draw on a vibrant research history in flexible visual exploration through the use of dynamic query concepts [3]. Grokker [www.grokker.com] is one commercial example notable for its use of circular groupings of dynamically-generated topic clusters on Web search results. With FacetMap, we wanted to see if we could maintain this flexible model of exploration without having to apply it only to search *results*, and without succumbing to size/interactivity limitations in the data domain. It has been reported that (as of 2002) state-of-the-art interactive visualizations usually are degraded to unusable by around 10,000 items [10].

In much of the above-cited work, the visualizations are also very domain-specific – meaning that the interesting attributes of the data, the primary axes of the visualization, are picked in advance, and the visualization is difficult (or impossible) to port to other datasets with different items or item types. One interesting exception is the Relation Browser++ [29]. This fifth iteration of a faceted metadata search/browse interface presents facets and their top-level values simultaneously, with embedded bar graphics to visualize the value distributions and mouse-over effects for query previews. Notably, it relies on a general purpose data format for wide portability, but in its display it is actually limited to a few simultaneous facets and a relatively small number of values per facet. With FacetMap, we explicitly set out to design an interface that scaled naturally with the size of the dataset while maintaining its portability across dramatically different data domains.

At the other end of the spectrum, high-end database exploration tools such as Spotfire [1] and Polaris [26] are domain-agnostic. They provide high levels of customization and control to the user, whom they assume to be an expert in the domain and proficient with the tool. In FacetMap we wanted to create a visualization with a very simple control surface that did not require the user to spend time designing attribute mappings or exploring visualization settings, even if that meant a certain loss of power in customization.

3 FACETMAP DESIGN

FacetMap allows users to perform search and browse tasks by iteratively selecting attributes in order to filter the dataset and refine the displayed set of results. The driving principle behind FacetMap is to show any size dataset in the most useful way, given the screen space constraints, the number of items, and the attributes of those items. As Marcia Bates put it [4], there is

> *a valuable aspect of human cognition that is unfortunately much ignored in the information system design world: people can recognize information they need very much more easily than they can recall it. The average person will recall (think up) only a fraction of the range of terms that are used to represent a concept or name, but can take in a screen full of variants in an instant, and make a quick decision about desired terms for a given search. Most current information systems require that the searcher generate and input everything wanted.*

Accordingly, rather than beginning with a blank results area, FacetMap begins by querying the store for all the facets that might serve as useful ways to organize items (by time, by type, by author, etc.), and tries to lay out each facet in 2D space with an appropriate view of the items inside (Figures 1 and 2) and a screen area for each facet proportional to its item count. This graphical affordance draws greater attention to facets with wider applicability – for example, *Date*, which might apply to every item in the data set, would have a relatively large screen area, whereas *Camera Type*, which might only apply to photograph items, would have a proportionally smaller space.

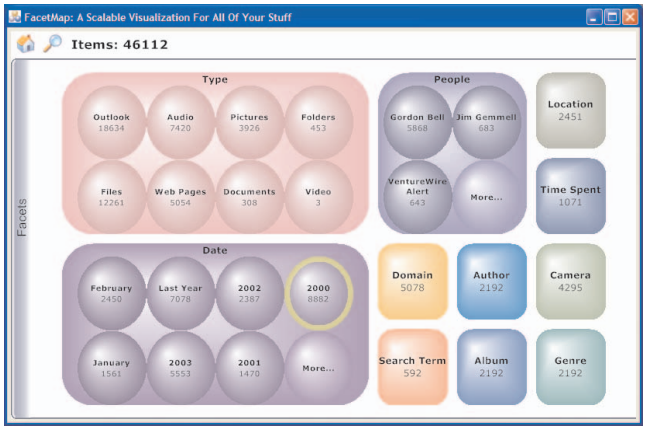

Fig. 2. In the purple Date facet at left bottom, the user is about to select the bubble for year 2000.

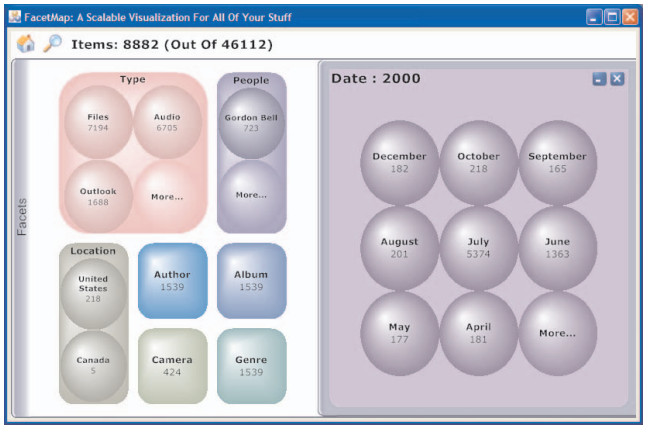

Fig. 3. After an animated transition, the dataset is now filtered to the 8882 items with a Date year attribute of 2000. The Date facet now appears in the active facet region on the right. Some facets that are no longer applicable have disappeared, and the remaining facets on the left (Type, Location, etc.) have been updated with new distributions.

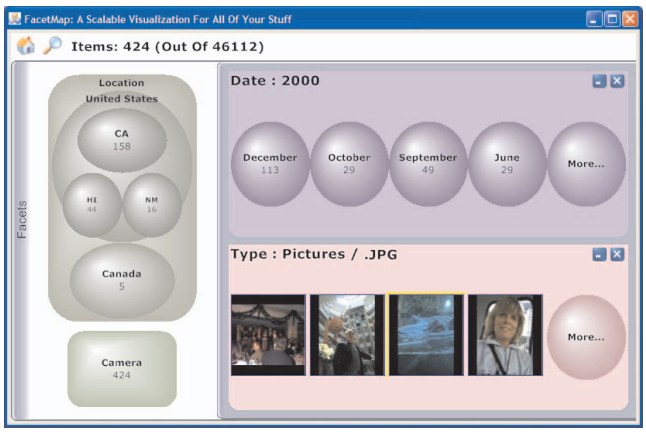

Fig. 4. The user has further narrowed down by Type/Pictures/JPGs and is now left with just the 424 jpg pictures taken in the year 2000. Several of those pictures are already visible within the Type facet region at lower right, while the upper right Date region offers month-level groupings for further date refinement. Two other facets remain on the left for further query refinement.

When the dataset is too large to display all items within each facet at a useful size, FacetMap collapses individual items into oval *bubbles* representing groups of items that share common attribute values. This offers users an overview of the distributions of items along multiple axes simultaneously, as well as a useful set of choices for narrowing the dataset down. When there are too many child bubbles to fit in a given sub-region, successive sets of off-screen groupings can be paged through by selecting the "More…" child bubble.

To perform a search/browse task, the user interactively selects a bubble within a facet and clicks on it. This applies a refining filter to the data, reducing the dataset to only those items matching the selected criteria. For example, if the user selects the "2000" bubble in the *Date* facet, only items with a year 2000 date attribute would remain. FacetMap animates away the eliminated items from every facet view and begins the layout process over again, this time devoting the screen solely to the remaining items and their available facets. A simple example of this sequence is shown in Figures 2, 3, and 4.

Overall screen space is divided into two sections: On the left are facets which have not yet been used in a filter clause ("remaining facets"), and on the right are facets with values currently being used in the overall filter ("active facets"). The dividing line between them moves proportionally leftward as the number of facets selected into the right section grows. (Either section disappears entirely when its membership is null.) The active facet section is further subdivided into equal height horizontal regions corresponding to the individual active facets, as shown for *Date* and *Type* in Figure 4. At the top of each active facet region is a title area showing the selected attribute values that define this facet's portion of the current overall filter, and offering control surfaces for manipulating the filter terms and the layout for the region (Figure 5). The remainder of each facet region contains a facet-specific view of the remaining items (as defined by the combined global filter of all active facet filters). When the user gets down to the most granular level of a given facet, the individual items are shown. The user can hover over an item to get a temporary popup view with more information, double-click on an item to open it, or continue to filter the dataset based on other facets and facet values.

Fig. 5. The title area of an active facet region for Date. It can be minimized (re-allocating its space to the other active facets) with the "minimize" (underscore) button. The entire filter clause can be eliminated with the "close" button, and the sub-clauses of the facet filter (e.g. Oct 1st) can be selectively eliminated by clicking on them. A strike-out hover effect is used to suggest this behavior.

By repeatedly selecting from among the displayed item groupings and the controls for eliminating or relaxing filters in various facets, the user can quickly narrow, expand or pivot the selection of filtered items and explore the structure of the dataset. The user always has multiple parallel views of the remaining items, facilitating discovery of the distributions and relationships among the items in the dataset.

We considered it very important to also include a free-text search capability in the overall system. FacetMap integrates conventional text-based search into the experience with the search button shown in Figure 6. Clicking this button produces a modal dialog box allowing the user to enter text. Submitting a text term adds a full-text filter clause to the overall global query and creates a corresponding new active filter region to represent the search term. The rest of the system responds just as it would with the addition of a new facet-based filter clause to the global query, allowing the user to see the facets and facet value distributions among the remaining items matching the new overall query.

3.1 Visual Design

In this section we describe the five guiding principles that drove our design. Although some of these design criteria may seem overly restrictive, they did lead us to take a radically different approach and to design a novel visualization and interaction technique.

Aesthetically-pleasing output: avoiding text lists and scrollbars for adaptive, graphical presentation. The facet bubbles were extremely visual and colorful by design. A single text label was provided for the top-level facets and the value bubbles inside, along with a numeric indicator of how many items were contained within. There were no scrollbars or text lists in the interface, and thumbnail views of individual items were used whenever available. The rounded aspect of the facets, the spatial layout, the colors, and the animations were all chosen with the hope that they would be pleasing to browse.

Output-is-input: emphasizing Shneiderman's concept of direct manipulation of on-screen objects for consistency and tight coupling [25]. Users interact with the facets and facet values by clicking directly on them, which reveals a further breakdown of clickable values, which eventually reveal in-place thumbnails of the individual items themselves. In addition, groupings and facets with no items are removed, meaning that the user can never encounter a null query. All interactions perform useful filtering and a query will never lead to an empty dataset. A simple animation is used to help the user understand that their choices reveal new options for interaction: in the first phase, bubbles that are being eliminated by the refinement are animated away. In the second phase, existing nodes are moved and scaled into new positions and sizes, and nodes representing new breakdowns grow into place.

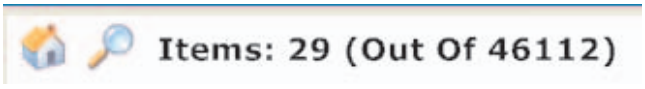

Fig. 6. The title area for the entire visualization continually displays the number of items currently in the visualization. The "home" button (house icon) clears all query clauses and returns to the initial global view. The "search" button (magnifying glass icon) allows the user to enter a text search term, and a new facet region representing this term then enters the visualization as a filter clause.

Simple interaction: using a single, consistent metaphor for all interactions, without multiple modes and control sets. We strove for a simple, "web-like" interaction based on single clicking the facets and facet values. With each selection, a filter clause is added to the set of global restrictions on the dataset, building what amounts to a complex database query purely with graphical manipulations. To "undo" a selection, the user simply clicks on the appropriate query sub-clause, or on the "close" button for the entire facet filter (Figure 5). A "home" button is provided at the left-hand side of the main visualization title area to clear the display of all filters and return to the initial view at any time (Figure 6).

Scalability: designing for scaling gracefully into arbitrary 2D display regions, across a wide range of dataset sizes, and over heterogeneous datasets with arbitrary facets (Figure 7). As described in further detail below, FacetMap uses a query-driven, space-filling algorithm (herein referred to as BubbleMap) to drive the graphical layout. As mentioned before, the relocations of facets, attributes, and items dictated by this dynamic layout, and the resulting broad transition animations, are a potentially serious usability concern. But spatial stability of interface elements is in inherent conflict with the ability of an interface to adapt to different distributions of items and attribute values among various facets. We specifically wanted to explore a data-driven layout that, unlike a hand-crafted design of

domain-specific metadata, would not be rendered obsolete by the application of the first filter clause or by evolution of the dataset itself.

Integrated browse and search: conveying a sense of structure in the corpus at every step of the interaction to guide exploration – both purposeful and serendipitous. The multiple parallel views in various facets allow users to visually scan for useful search clauses while simultaneously giving an overview of the various item attribute distributions. A text search facet can be invoked at any time during the interaction for a more targeted dive, but in those occasions where there are still too many matching items the visualization will automatically offer further refinement options among the remaining applicable facets.

Fig. 7. FacetMap scaling to a 9-panel LCD display.

3.2 System Implementation

FacetMap is a Windows executable written in C#. We used the .NET Framework's built-in class support for data access and for graphics (GDI+), and we used the University of Maryland's Piccolo.NET library for scene graph and animation support. FacetMap is designed to dynamically query a Structured Query Language (SQL) database in order to produce its visualizations. Although the current implementation runs on top of a MyLifeBits personal data store [12], FacetMap was written expressly to generalize to nearly any SQL-based faceted dataset, and so the dataset requirements can be described in fairly simple, abstract terms. FacetMap relies on a logical set of queryable "facet tables" with the generic form <ItemID><FacetValue>. Dataset items appear one or more times in every facet table for which they have a corresponding attribute value. For instance, an email item may not have a "Location" attribute value, so there would be no row in the "Location" facet table for that email ItemID, but it may have multiple rows in the "People" facet table, one for each person listed on the To: line of the email. FacetMap simply requires that, when queried within the context of a specific filter criterion, each facet table be able to return upon demand: a) a count of items remaining[1], b) the groupings of facet values among the remaining items[2], and c) the unique item IDs representing the set of remaining items[3].

An item's logical FacetValue may be represented in multiple columns with different granularities to support hierarchical facets – for example, the "Location" facet table may have a "Country," "State," and "City" column breakdown. FacetMap reads the descriptions of the facet tables and their column names for a given dataset from a master facet table on startup, and builds a corresponding list of facet classes internally to be used in generating queries. The facet descriptions include a name, a distinctive color, and the data type of the constituent attribute values.

Within the remaining facets section of the visualization, and within any given active facet region, the BubbleMap algorithm is applied to automatically generate the layout of facets and their values. BubbleMap was inspired by TreeMaps [16], specifically the Quantum Pivot TreeMap (QPT) described in [5]. BubbleMap has two main differences with the QPT. First, in its graphical output it uses rounded rectangles (at the top level) and ovals (at lower levels) instead of rectangles. This requires various new scaling factors to be introduced into the base algorithm during recursive descent. Second, BubbleMap introduces dynamic cutoffs to the algorithm's descent to preserve upper limits on the information density of the output. These two variations create a space-filling visualization that uses whitespace to make groupings and structural relationships in the layout more easily perceived than with standard TreeMaps.

BubbleMap first determines how many nodes N would fit in a grid layout according to internal constants representing the aesthetically-appropriate minimum node size (currently set at approximately 1 square inch). FacetMap then dynamically generates an SQL query designed to return the top N+1 facet value groups and their item counts for the current facet. Additional WHERE clause restrictions are added to this query to represent the filter clauses in effect among the active facets of the global query (if any), and the query is submitted to the database. If N or more values are returned from this query, or if there are no more value groups but only individual dataset items remaining, they are laid out in a simple grid in the given region (with the addition of a "More…" bubble to represent values or items after the first N nodes). If fewer than N grouping values are returned, they are submitted to BubbleMap's QPT algorithm, using the returned values' item counts as the relative TreeMap "area" metrics. If this algorithm is able to successfully calculate expanded sub-areas for individual facet grouping values to contain further detail (subject to the aforementioned limits on aesthetics and minimum node size), each such sub-area becomes the target of recursive BubbleMap application. In this recursive descent, the facet value representing the sub-area becomes an additional filter clause in the dynamically-generated SQL statements, and the value groupings within it are requested at the next finer level of granularity. In effect, FacetMap is performing a recursive series of "pre-queries" across the entire facet space on behalf of the user, automatically devoting more detailed queries (and screen space) to the areas of the attribute space with more items in them.

Note that the number of queries generated by FacetMap to fill a display grows linearly with the display area, and is independent of the dataset size. Also, the queries that FacetMap generates (specifically, TOP-N, count and grouping queries) can be satisfied efficiently by the database's indexes without recourse to scanning the source tables. These two factors combine to allow FacetMap to scale gracefully in the data layer.

4 User Study

We conducted an exploratory user study to benchmark FacetMap against the Memex front-end to MyLifeBits [12], as well as to examine the success of our design with regard to users' mental models of faceted metadata search. This user study was intended as a formative evaluation, primarily to inform further iterations of the system. The choice of comparison system among existing faceted and text-based query engines across many different domains was difficult; ultimately, we felt Memex – as a domain-optimized, faceted, text-oriented query system – would provide an initial, isolated test of FacetMap's graphical approach to information retrieval, on top of the exact same underlying data (namely, a large subset of Gordon Bell's personal information store), and with future comparisons we could move on to more established competition. In the Memex interface (see Figure 8), several facets are arranged in

[1] SELECT COUNT (DISTINCT *item_id*) FROM *<facet_table >*
[2] SELECT *<facet_column>*, COUNT(*) AS Total FROM *<facet_table>* GROUP BY *<facet_column >*
[3] SELECT *<item_id>* FROM *<facet_table>*

tabs along the left-hand side of the interface, one of which can be expanded at any moment to offer facet-specific filter possibilities. The cumulative set of applied filters is shown in a box at the upper left, and free-text search terms can also be typed into this box at any time. The bulk of the interface window is taken up by the results region, a large scrollable list box of text or thumbnails. In order to benchmark FacetMap performance, we examined the differences between the two user interfaces for targeted keyword search tasks (where a key text fragment could be used to isolate a single appropriate answer) versus attribute browsing tasks (where various attributes were needed to narrow down a large set to a small one). We hypothesized that the Memex user interface would be difficult to beat for more textual, targeted searches, but hoped that FacetMap would not suffer by comparison and would perform better as the search criteria became more exploratory. We limited FacetMap to Memex's fixed display size (1000x700 pixels), negating the benefits of FacetMap's scalability but providing a stricter test of its usability as a search tool.

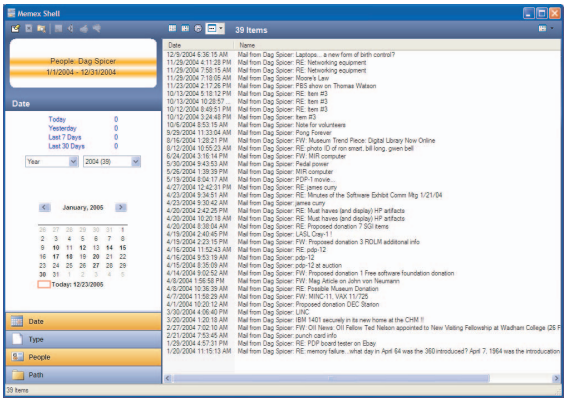

Fig. 8. Screenshot of the Memex interface.

Since the database consisted of Gordon Bell's digital information (his email, frequented web sites, documents, photos, etc.) from as early as the 1930's, all tasks involved looking up information Gordon had saved. Example tasks included:

Targeted: Find the earliest piece of email Gordon received from Jim Gemmell (text search for "Gemmell").

Browse: Name a document that Gordon modified in the 3rd week of May, 2000.

4.1 Participants

We recruited 10 participants (5 female) for this study. All were intermediate to expert users of the web and used search tools for a minimum of one hour a week. Users ranged from 20 to 58 years of age (average 31.5), and had used a computer for 15.1 years, on average. The study took less than 1.5 hours and users were given a software gratuity for their participation.

4.2 Methods and Procedure

The study was a 2 (user interface: FacetMap v. Memex) x 2 (search type: targeted v. browse), within-subjects, repeated measures design. Participants were asked to perform 8 search tasks (1 targeted and 7 browse) with one user interface, fill out a user satisfaction questionnaire, and then perform a different set of isomorphic tasks (again 1 targeted and 7 browse) with the other user interface. User interface (FacetMap or Memex) and task type were fully counterbalanced between participants.

4.3 Results

4.3.1 Task Times and Satisfaction Data

We submitted the average task times to a 2 (user interface) x 2 (search type) Analysis of Variance (ANOVA). Results showed a significant effect for user interface, F(1,9)=7.06, p=.026, with the Memex UI faster than FacetMap overall (258 v. 274 seconds, on average, respectively). In addition, there was, as suspected, a significant interaction for the benefits of using each interface depending on the search task type, F(1,9)=22.26, p=.001. Pairwise comparisons using the Bonferrroni correction for multiple tests showed that Memex was significantly faster than the FacetMap UI for targeted search at the p=.05 level. Neither user interface was significantly faster than the other for browse tasks (Figure 9).

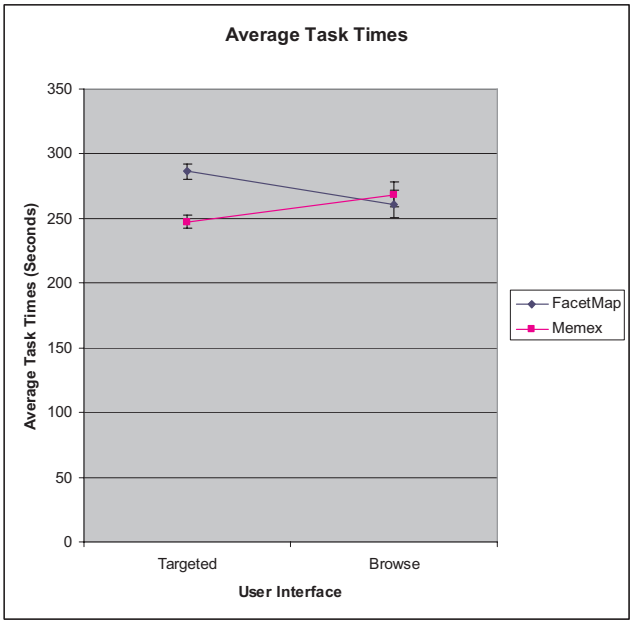

Fig. 9. Average task times by search type (targeted v. browse). Error bars are +/- 1 SE.

We also analyzed user satisfaction ratings for the two different interfaces. We observed only two borderline differences in the ratings. Memex was rated faster in terms of perceived system response time (average rating of 5.7 v. 4.8 for FacetMap), t(18)=-1.6, p=.1. In terms of aesthetic appeal, FacetMap scored higher than Memex (average rating of 5.3 v. 4.1), t(18)=1.9, p=.07. All of the average user satisfaction ratings are provided in Table 1.

Table 1. Average User Satisfaction Ratings (Standard Deviations in Parentheses)

Question	FacetMap	Memex
Mental Demand	4.0 (1.8)	4.3 (1.6)
Physical Demand	3.6 (2.1)	3.6 (1.6)
System Response Time	4.8 (1.4)	5.7 (1.1)
Satisfaction	5.6 (1.4)	5.4 (0.8)
Preference over Existing Techniques	4.9 (1.2)	5.2 (1.4)
Browsing Support	5.9 (0.9)	5.9 (0.9)
Text Search Support	5.9 (1.4)	5.3 (0.8)
Aesthetic Appeal	5.3 (1.3)	4.1 (1.5)

Overall, four users preferred FacetMap, five users preferred Memex, and one user said he could not choose between them. There was a near perfect correlation between preferring the more graphical user interface, FacetMap, and whether or not the user viewed their

photographs in thumbnail view in Windows Explorer. Several participants recognized that a combination of the two user interfaces might be a nice alternative design, or that allowing either view might work best for the general population.

4.3.2 Usability Observations

We observed usability issues with FacetMap which we believe to be potential concerns for any graphical, faceted search tool.

Choosing the "Right" Facets: Interestingly, though we did see some initial user concern over knowing which facet to select when initiating a search, users were surprisingly successful with a variety of strategies because the faceted breakdown offered multiple paths to success. For instance, one user chose *Camera Type* as a facet when she was looking for a picture (instead of the more obvious *Type:Picture*) but she succeeded because the results of the two queries were identical. Some users felt that the order they applied filters mattered a great deal (for instance, one user always chose the *People* facet before using text search for a target's last name). While it did not prevent them from performing the search, this misconception was slightly restricting in their interactions. We are now considering augmentations which make it clearer that the order in which filters are applied does not matter. Related to this, users did figure out how to delete filters and essentially go "back", but a few asked that an explicit back button be provided, a change worth considering.

Search for Facet Values: Occasionally, we observed suboptimal search strategy when the user had determined to locate a particular value in a particular facet, but the facet's hierarchy had too many groupings at one level, requiring a large number of iterations with the "More…" bubble. Also, some users thought that they could search for facet values via full-text search. For example, if asked to find a particular email in the database, they would perform a text search for "email", rather than looking in the *Type* facet for *Email*. There was clearly confusion in these users' minds between facet values and content keywords. These usability issues suggested to us that we need to develop sub-controls for rapidly searching the value space within a facet for specific, anticipated values. We plan on implementing two features to address this. One is the ability to dynamically reorder the bubbles within a facet (such as by count, alphabetically by label, by usage, or by some facet-specific ordering) to enable more efficient scanning of the values. The other feature is an incremental text-entry element on each bubble that would perform dynamic partial-match text search of facet value labels. This would allow a scoped search of a facet's child bubbles and rapid invocation of specific filter clauses buried deep in the faceted metadata hierarchy. Whether (and how much) to include metadata in the full-text search of a faceted search system is a familiar, difficult decision, and when the metadata *is* included it creates another set of interface difficulties in showing *why* a particular item matched. For example, searching for "Thursday" could return all email with the word "Thursday" in the text, as well as all email sent on a Thursday. Our hope is that by providing text input specifically scoped to metadata, we can provide text search of facet values without compromising the usability of full-text search across item contents.

Graphical Approach: Occasionally we observed participants reveal search targets but not recognize them during subsequent visual search. Some of these users indicated that linear lists might have helped them locate the items more quickly and reliably. However, several users also commented favorably on the graphical design of the interface. In consequence, we are considering ways to integrate short lists into the overall graphical model of the visualization, especially once several filters have been engaged. Also, one user suggested we allow multiple-selection of several facet values at once to reduce interaction time. This straightforward change could easily be incorporated in subsequent iterations.

4.4 Study Discussion

We designed the tasks and conditions in the study so as not to take specific advantage of the scalability and generality features of FacetMap. The TreeMap-based interface is designed as much for corpus understanding and overview as for targeted search, yet we did not test for these benefits. This is because we felt that a pure browsing interface that was unusable or deficient in rapid, targeted search was not a worthwhile or novel goal. While overall completion times for FacetMap were slower on the tasks tested, it is interesting and encouraging to us that the more visual and unusual search user interface fared as well as it did when compared to a more conventional, domain-optimized, list-oriented UI. The graphical novelty and dynamic space allocation did not appear to be detrimental (as one might have expected), either in subjective preference or in rapid search performance. Several excellent ideas were provided by participants for improving the design and usability of FacetMap, and we are carefully considering these ideas as noted above. Our next effort will be to deploy a new iteration of FacetMap so that users can use it to search their own desktop databases as an alternative to today's text-oriented desktop search tools. Our goal is to study search behaviors using graphical facets over a much longer period of time, on users' real documents and data.

5 Conclusion

This paper described a graphical alternative to the dominant text-based approach to searching and browsing large data stores, particularly for personal information. FacetMap uses a space-filling visual metaphor for both output and input, by allowing the user to select graphically displayed metadata facet values as filters. As filtering facet values are selected, the display space is divided to show parallel facet-specific views of the current resulting item set, as well as the remaining relevant facets available for further filtering. A user study was performed to compare this graphical approach to a more traditional, text-oriented, faceted search system. Results of the study suggest that FacetMap has more work to do to reach parity with the more traditional approach for targeted searching tasks, but we have some concrete ideas for how to get there. Meanwhile, the graphical approach is already comparable in performance for browsing tasks, where specific search details are less well specified. We have also presented scalability characteristics, in both the visual and data domains, that suggest our approach will remain viable as displays and datasets both continue to increase in size. This leads us to believe that we can create an aesthetically-pleasing, query-driven visualization for a wide range of personal, metadata-rich data stores that offers interactive searching and browsing to serve a wide range of user tasks.

Acknowledgements

We would like to thank Jim Gemmell, Roger Lueder, and Gordon Bell for many insights into the problem space, and for providing the data repository for research and study.

References

[1] Ahlberg, C., Spotfire: an information exploration environment, *ACM SIGMOD Record, 25,* 4, pp. 25-29, Dec. 1996.

[2] Ahlberg, C. and Shneiderman, B., Visual information seeking: tight coupling of dynamic query filters with starfield displays, *Proc. CHI '94*, pp. 313-317, 1994.

[3] Ahlberg, C., Williamson, C., and Shneiderman, B., Dynamic queries for information exploration: An implementation and evaluation, *Proc. CHI '92*, pp. 619-626, 1992.

[4] Bates, M., Indexing and access for digital libraries and the internet: Human, database, and domain factors, *JASIS, 49*, pp. 1185-1205, Nov. 1998.

[5] Bederson, B., Shneiderman, B., Wattenberg, M., Ordered and quantum treemaps: Making effective use of 2D space to display hierarchies, *ACM Transactions on Graphics, 21*, 4, pp.833-854, October 2002.

[6] Bush, V., As we may think, *Atlantic Monthly*, July, 1945.

[7] Cutrell, E., Robbins, C., Dumais, S., and Sarin, R., Fast, flexible filtering with Phlat – Personal search and organization made easy, to appear in *Proc. CHI '06*, 2006.
[8] Cutting, D., Karger, D., Pedersen, J., and Tukey, J., Scatter/gather: A cluster-based approach to browsing large document collections, *Proc. SIGIR'92*, pp. 126-134, 1992.
[9] Fabrikant, S., Evaluating the usability of the scale metaphor for querying semantic spaces, *COSIT 2001*, pp. 156-172.
[10] Fekete, J., Plaisant, C., Interactive information visualization of a million items, *Proc. InfoVis '02*, p. 117, 2002.
[11] Furuta, R., Shipman, F., Marshall, C., Brenner, D., and Hsieh, H., Hypertext paths and the world-wide web: Experiences with walden's paths, *Proc. Hypertext '97*, pp. 167-176, 1997.
[12] Gemmell, J., Bell, G., Lueder, R., Drucker, S., and Wong, C., MyLifeBits: Fulfilling the Memex vision, *Proc. Multimedia '02*, pp. 235-238, 2002.
[13] Hetzler, B., Whitney, P., Martucci, L., and Thomas, J., Multi-faceted insight through interoperable visual information analysis paradigms, *Proc. InfoVis '98*, pp. 137-144.
[14] Hull, J., Hart, P., Toward zero-effort personal document management, *Computer*, *34*, 3, pp.30-35, March 2001
[15] Jerding, D. and Stasko, J. The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. *IEEE Transactions on Visualization and Computer Graphics*, pp. 257–271, 1998.
[16] Johnson, B. and Shneiderman, B., TreeMaps: A space-filling approach to the visualization of hierarchical information structures, *Proc. Visualization '91*, pp. 284-291, 1991.
[17] Lamming, M. and Flynn, M., "Forget-me-Not": Intimate computing in support of human memory, *Proc. FRIEND21 Symposium on Next Generation Human Interface*, Feb 1994.
[18] Pirolli, P., Schank, P., Hearst, M., Diehl, C., Scatter/gather browsing communicates the topic structure of a very large text collection, *Proc. CHI '96*, pp.213-220, 1996.
[19] Plaisant, C., Milash, B., Rose, A., Widoff, S., and Shneidermann, B., Life Lines: visualizing personal histories, Proc. CHI '96, pp. 221-227, 1996.
[20] Rekimoto, J., Time-machine computing: A time-centric approach for the information environment, *Proc. UIST '99*, pp. 45-54, 1999.
[21] Rennison, E., Galaxy of news: an approach to visualizing and understanding expansive news landscapes. In *Proc. UIST '94*, 1994.
[22] Rhodes, B. and Starner, T., The remembrance agent, *Proc. PAAM '96*, pp. 487-495, April 1996.
[23] Ringel, M., Cutrell, E., Dumais, S., and Horvitz, E., Milestones in time: The value of landmarks in retrieving information from personal stores, *Proc. INTERACT '03*, pp. 184-191, Sept. 2003.
[24] Sanderson, M., Croft, B., Deriving concept hierarchies from text, *Proc. SIGIR '99*, pp. 206-213, 1999.
[25] Shneiderman, B., Direct manipulation: a step beyond programming languages, *Computer, 16*, 8, pp. 57-60, Aug. 1983.
[26] Stolte, C., Tang, D., and Hanrahan, P., Polaris: a system for query, analysis, and visualization of multidimensional relational databases, *Transactions on Visualization and Computer Graphics, 8,* 1, pp. 52-65, January 2002.
[27] Yee, P., Swearingen, K., Li, K., and Hearst, M., Faceted metadata for image search and browsing, *Proc. CHI '03*, pp. 401-408, 2003.
[28] Zeng, H., He, Q., Chen, Z., Ma, W., Ma, J., Learning to cluster web search results, *Proc. SIGIR '04*, 2004.
[29] Zhang, J., and Marchionini, G., Evaluation and evolution of a browse and search interface: Relation Browser++. *The National Conference on Digital Government Research*, 2005.
[30] Zhang, J., Mostafa, J., and Tripath, H. Information retrieval by semantic analysis and visualization of concept space for the D-lib magazine. *D-lib online magazine*, 2002.