Magnet: Supporting Navigation in Semistructured Data Environments

Vineet Sinha
vineet@csail.mit.edu

David R. Karger
karger@theory.lcs.mit.edu

MIT Computer Science and Artificial Intelligence Laboratory (CSAIL)
32 Vassar Street, Cambridge, MA 02139

ABSTRACT

With the growing importance of systems containing arbitrary semi-structured relationships, the need for supporting users searching in such repositories has grown. Currently support for users' search needs either has required domain-specific user interfaces or has required users to be schema experts. We have developed a general-purpose tool that offers users helpful navigation and refinement options for seeking information in these semistructured repositories. We show how a tool can be built without requiring domain-specific assumptions about the information being explored. In addition to describing a general approach to the problem, we provide a set of natural, general-purpose refinement tactics, many generalized from past work on textual information retrieval.

Keywords

Navigation, searching/browsing, information retrieval, semistructured data, metadata

Categories and Subject Descriptors

H.3 [**Information Search**]: Information Search and Retrieval;
H.3.3 [**Information Search and Retrieval**]: Query formulation;
H.4.3 [**Communications Applications**]: Information browsers

1. INTRODUCTION

Modern digital information takes many forms: e-mails, appointments, papers, online purchases, etc. These forms of information often have *attributes*—for example, e-mails have subjects, appointments have meeting times, and online purchases have prices. Additionally, the information is connected by *relations* of various types to other pieces of information—for example, an e-mail sender's contact information, an appointment location, and online purchase sites. The use of attributes and relations (structured information) allows for increased precision in information management activities, compared with typical text-based (unstructured) information. For example, while users of unstructured text-search systems are able to search for books *containing* the word "James," systems leveraging the structure available in metadata allow users to distinguish in their searches between books *about* James and books *written by* James.

In such systems leveraging structured data, a key challenge is to support the search process. Currently support for powerful database searches is available *if* the user is a sophisticated database administrator who is familiar with the schema, the data, and the system's querying functionality. Alternatively, one can sometimes build good naive-user interfaces in a domain-specific manner and on a per-schema basis. Building a search interface is difficult because users frequently do not know exactly what to ask the system. In such cases they often seek to take an iterative approach. They need to be able to repeatedly pose queries, review the results, and refine the query, to "home in" on the target information [2, 22]. Easy-to-use support for powerful searches over varying schemata requires that we offer this kind of iterative browsing and refinement.

In this paper, we present Magnet, a system supporting naive-user navigation of structured information via a domain-independent search framework and user interface. We show that the vector space model popular in textual information retrieval can be effectively adapted to structured information, *without* sacrificing the benefits of structured information over unstructured. Doing so lets us take several fuzzy navigation and browsing techniques popularized in the text-retrieval domain and apply them to structured information. Magnet supports both database-oriented refinement based on specifying additional attributes, and refinement based on fuzzier notions of "similarity." Magnet is able to use these techniques to support browsing even in the absence of any schematic information. At the same time, it takes advantage of any available schematic information to specialize and thus improve the user's browsing interface.

Because Magnet does not rely on carefully crafted schemas or other domain knowledge, it can be applied to *semistructured* data just as easily as structured data. Magnet consumes RDF [18], a WWW standard for representing semistructured (and structured) data as a semantic network, connecting items with labeled arcs. Magnet's methods can apply equally well to XML [29], another popular representation of semistructured information. We foresee a significant growth in both the availability and the number of contributors of such semistructured information. This will happen at the enterprise level as part of the growth of the Semantic Web. It may also happen at the individual level, if tools like RDF help in the long-overdue migration of individual users' structured information from application-specific repositories to general-purpose databases. As semistructured data becomes more common, tools for browsing it become more important. And as the pool of contributors creating semistructured data grows, we will face an explosion in the number of different (and possibly incomplete or inaccurate) schemas under which information is disseminated. In such a situation, it becomes less and less feasible to design a domain-specific interface for each new schema encountered: instead, we need tools such as Magnet that can provide effective browsing of newly encountered, or less than fully structured, information.

At the same time, it is beneficial to take advantage of whatever schema information *is* available. We discuss certain useful schema annotations and show how Magnet makes use of them, when available, to improve users' navigation experiences by specializing the available browsing steps and the way they are presented to the user.

2. RELATED WORK

A wide variety of work has been done in developing end-user interfaces to support sophisticated and precise information needs. For unstructured text environments, Scatter/Gather offers a navigation system based on document clustering [6]. Given a large collection, Scatter/Gather creates topical clusters and lets the user pick ones that seem interesting to create a smaller collection. Similarly, the NaviQue prototype provides a single zoomable visualization for users to query and view results in an attempt to blend the ease-of-use of browsing interfaces with the power available from querying [10]. For a given collection, NaviQue uses document-similarity to represent distances between documents in the visualization, with clusters of documents having a topic displayed. Users are allowed to refine the collection by either entering keywords, or selecting an interesting cluster of documents and zooming in on the cluster. Scatter/Gather and NaviQue demonstrate the synergies that can be achieved by supporting navigation and querying together, and Magnet tries to achieve similar synergies in structured models.

Among systems trying to leverage the strength of structured models, Catarci et al. [5] have surveyed a large number of visual query systems. These systems are designed for databases and present interfaces on top of domain-specific schemas known in advance. Similarly, the Flamenco project [28] provides powerful tools to help users refine queries by selecting metadata, and has thus demonstrated the power of content-oriented category metadata. While Flamenco requires a schema expert to provide the domain-specific customizations, Magnet focuses on being applicable in the absence of such information (with hooks to support customization to the level of other systems). Magnet tries to be agnostic about the data being browsed; it inspects existing data and attempts to automatically provide interfaces similar to those handcrafted for Flamenco.

A number of systems have been designed to interact with semistructured data; however, such systems typically focus on displaying the semistructured graph and performing queries, rather than supporting user searches with browsing and refinement of query results. Lore uses a concept called Dataguides to retrieve structural schema summaries and uses the summaries to support query formulation by allowing users to fill in constraints in one or more of the many possible fields [19]. While the use of Dataguides improves the search process, users need to explicitly perform queries on the system to browse results. Interactive querying with Dataguides [11] was supported by improving system responsiveness and building structural schema summaries on the search results. That system tries to ease the process of the user explicitly querying against a graph, but does not provide users with suggestions for their next step (to make the search process intuitive and incremental). Trigoni [26] uses an approach to support search strategies more explicitly. She allows the user to combine a set of templates describing the interesting portion of the semistructured graph. While templates are provided to effectively support various search strategies, the system makes users explicitly decide on one of the varying search strategies to combine in an SQL like query language.

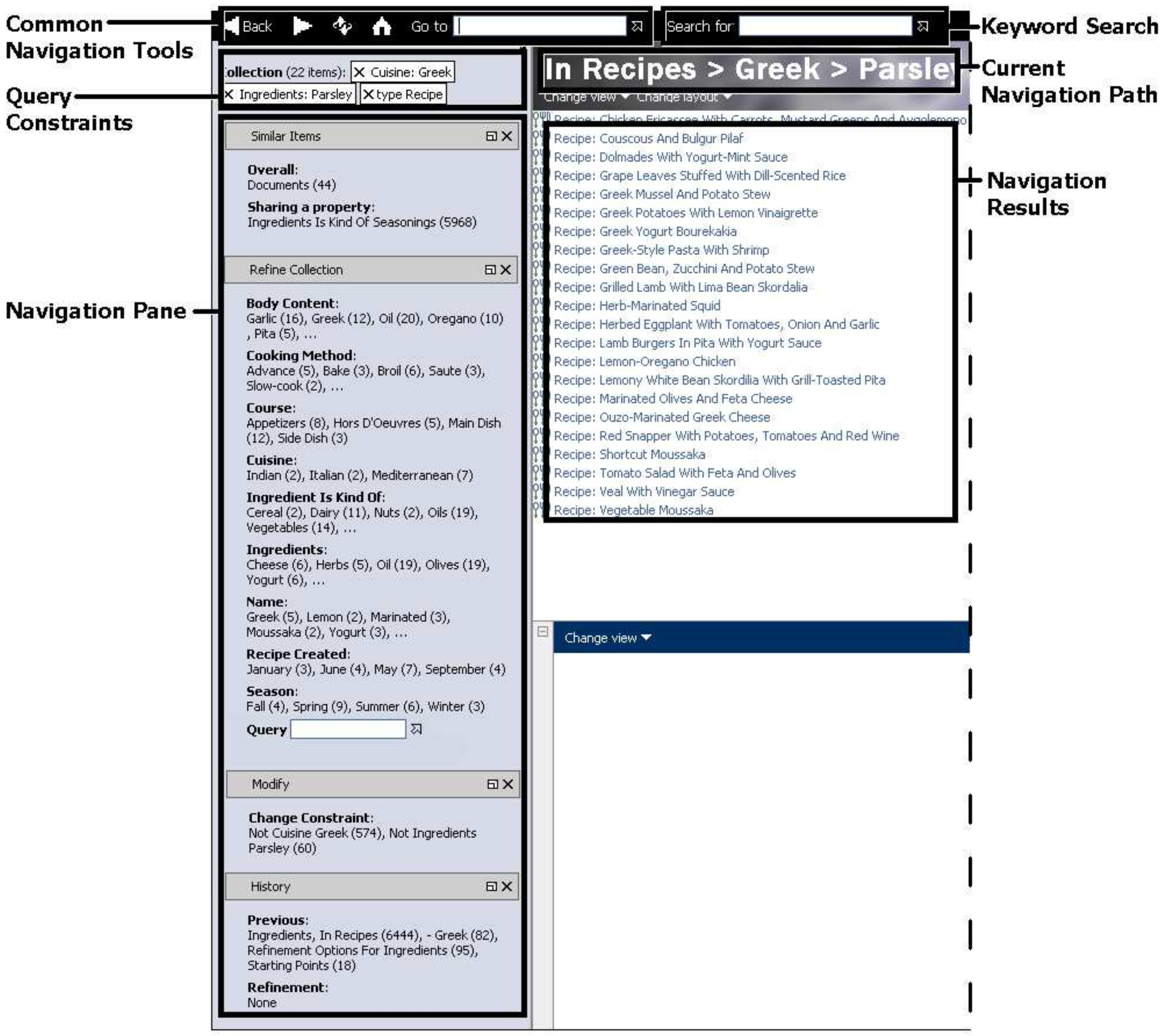
Other approaches to supporting searches in semistructured information have been used with XML documents. Carmel et al. [4] use contextual information found in the path to a desired document as terms for reranking documents and improving result sets. In contrast, Magnet focuses on offering the user summaries of result sets, to help users build up a better mental model of the data for better searching, and offers useful refinement steps that they can use to iteratively home in on their desired results. Egnor et al.[7] have attempted to use iterative refinement of structural information for searching by users who are not schema experts. While their approach has similar goals to that of Magnet, they use a starting keyword search to select a schema, and then limit the documents being searched to this schema, using schema information for refining their query. In contrast, Magnet supports users working with multiple schemata at any point in the search, and integrates all the refinement options based on any of the relevant schemata.

Magnet consumes RDF, an evolving WWW Consortium Standard for representing semistructured information [18]. RDF represents information by a directed graph in which the information objects are represented by nodes and the various attributes and relationships are represented by *property* links connecting the nodes to values (i.e., numbers and strings) or to other complex information objects. This type of semantic network representation is increasing in popularity. Another standard for storing semistructured data, XML, generally represents semistructured data as trees (rather than general graph), although it can use indirection to represent more general structures. There are often natural mappings from RDF to XML and back. Our approach is applicable to either (or any other) semantic network data representation, although our implementation is based on RDF. The Longwell suite [23] also provides navigation capabilities for RDF repositories; however, like the Flamenco project it also expects domain-specific customizations.

3. INTERFACE WALKTHROUGH

Magnet is a component of the Haystack's system [14]. To Haystack's general-purpose user interface for browsing and manipulating arbitrary semistructured information, Magnet adds the ability to browse towards an information need. Haystack provides a simple single-window browser-like interface for the user (shown in Figure 1), as well as panes for storing bookmarks, a scrapbook, starting points, etc. While Magnet is built on top of a semistructured repository, it uses an interface similar to the faceted browsing interface tested in the Flamenco project [28]. On top of this interface, Magnet presents the user with additional navigation options and is able to work seamlessly on arbitrary data sources.

Figure 1 shows the result of navigation in the repository consisting of recipes from the website Epicurious.com, used for the evaluation of our interface in a user-study. Like Lore's Dataguides the interface shows that the collection of recipes has properties like cooking method, cuisine type, and ingredient. Additionally, Magnet's interface also provides the users with an overview of the collection and shows the user that a large number of the recipes have cloves, garlic, olives and oil as ingredients. Magnet's interface further supports finding other recipes having similar ingredients, or those recipes sharing a common ingredient, and at the same time



Collection (22 items): ✕ Cuisine: Greek ✕ Ingredients: Parsley ✕ type Recipe

In Recipes > Greek > Parsley

Recipe: Chicken Fricassee With Carrots, Mustard Greens And Avgolemono
Recipe: Couscous And Bulgur Pilaf
Recipe: Dolmades With Yogurt-Mint Sauce
Recipe: Grape Leaves Stuffed With Dill-Scented Rice
Recipe: Greek Mussel And Potato Stew
Recipe: Greek Potatoes With Lemon Vinaigrette
Recipe: Greek Yogurt Bourekakia
Recipe: Greek-Style Pasta With Shrimp
Recipe: Green Bean, Zucchini And Potato Stew
Recipe: Grilled Lamb With Lima Bean Skordalia
Recipe: Herb-Marinated Squid
Recipe: Herbed Eggplant With Tomatoes, Onion And Garlic
Recipe: Lamb Burgers In Pita With Yogurt Sauce
Recipe: Lemon-Oregano Chicken
Recipe: Lemony White Bean Skordilia With Grill-Toasted Pita
Recipe: Marinated Olives And Feta Cheese
Recipe: Ouzo-Marinated Greek Cheese
Recipe: Red Snapper With Potatoes, Tomatoes And Red Wine
Recipe: Shortcut Moussaka
Recipe: Tomato Salad With Feta And Olives
Recipe: Veal With Vinegar Sauce
Recipe: Vegetable Moussaka

Similar Items

Overall:
Documents (44)

Sharing a property:
Ingredients Is Kind Of Seasonings (5968)

Refine Collection

Body Content:
Garlic (16), Greek (12), Oil (20), Oregano (10), Pita (5), ...

Cooking Method:
Advance (5), Bake (3), Broil (6), Saute (3), Slow-cook (2), ...

Course:
Appetizers (8), Hors D'Oeuvres (5), Main Dish (12), Side Dish (3)

Cuisine:
Indian (2), Italian (2), Mediterranean (7)

Ingredient Is Kind Of:
Cereal (2), Dairy (11), Nuts (2), Oils (19), Vegetables (14), ...

Ingredients:
Cheese (6), Herbs (5), Oil (19), Olives (19), Yogurt (6), ...

Name:
Greek (5), Lemon (2), Marinated (3), Moussaka (2), Yogurt (3), ...

Recipe Created:
January (3), June (4), May (7), September (4)

Season:
Fall (4), Spring (9), Summer (6), Winter (3)

Query

Modify

Change Constraint:
Not Cuisine Greek (574), Not Ingredients Parsley (60)

History

Previous:
Ingredients, In Recipes (6444), - Greek (82), Refinement Options For Ingredients (95), Starting Points (18)

Refinement:
None

Figure 1: The navigation system on a collection of recipes.

keeps track of the user's history.

3.1 Starting information searches

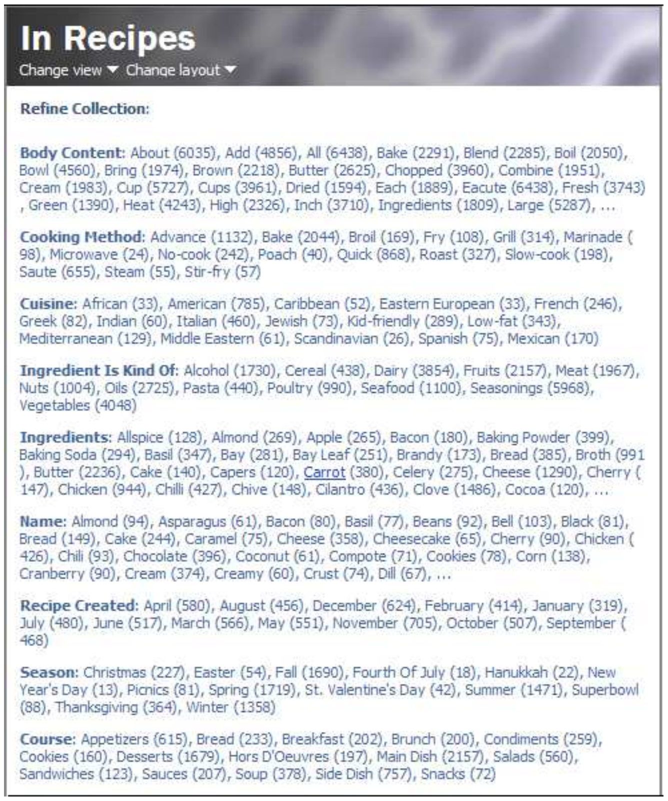
We expect that a search may often be initiated by specifying keywords, as this requires the least cognitive effort in planning the query. Users can do so by entering keywords in the toolbar shown in Figure 1. Keyword searches and other navigation steps often result in a collection of results as can be seen in the figure. Navigation suggestions are presented to the user in the navigation pane as can be seen in the left part of Figure 1.

Users arriving at large collections, were the navigation pane is inadequate, can use a specialized interface in the main pane (shown in Figure 2) to get a broad overview of the occurrence of metadata in the collection as well as to enable multiple navigation paths for refining the collection. While the number of navigation suggestions in this initial view of information may be large, the view provides an organized and sorted display of information to allow the user to gain a summary of the data and start the browsing session.

The interfaces shown in Figure 1 and Figure 2, jointly build dynamically the view for faceted metadata as suggested by Yee et al. [28]. Users can click and select a refinement option, such as Greek cuisine, to be presented with a collection of results in the main window and navigational suggestions in the left pane (similar to Figure 1).

3.2 Navigation pane

The navigation pane on the left in Figure 1 starts by showing at the top that the current collection is being displayed as the result of a conjunctive query consisting of three terms or constraints, i.e. the type of items are recipes, the recipes are of the Greek cuisine, and have an ingredient being parsley. The interface allows the user to further remove query terms by clicking on the 'X' by the constraints, and offers more powerful features like negation of the constraints through context menus (right clicking on the constraints). Thus, the user in Figure 1 can decide to either view all the Greek recipes (by removing the parsley ingredient constraint) or view all recipes containing parsley but those that are not Greek.



Refine Collection:

Body Content: About (6035), Add (4856), All (6438), Bake (2291), Blend (2285), Boil (2050), Bowl (4560), Bring (1974), Brown (2218), Butter (2625), Chopped (3960), Combine (1951), Cream (1983), Cup (5727), Cups (3961), Dried (1594), Each (1889), Eacute (6438), Fresh (3743), Green (1390), Heat (4243), High (2326), Inch (3710), Ingredients (1809), Large (5287), ...

Cooking Method: Advance (1132), Bake (2044), Broil (169), Fry (108), Grill (314), Marinade (98), Microwave (24), No-cook (242), Poach (40), Quick (868), Roast (327), Slow-cook (198), Saute (655), Steam (55), Stir-fry (57)

Cuisine: African (33), American (785), Caribbean (52), Eastern European (33), French (246), Greek (82), Indian (60), Italian (460), Jewish (73), Kid-friendly (289), Low-fat (343), Mediterranean (129), Middle Eastern (61), Scandinavian (26), Spanish (75), Mexican (170)

Ingredient Is Kind Of: Alcohol (1730), Cereal (438), Dairy (3854), Fruits (2157), Meat (1967), Nuts (1004), Oils (2725), Pasta (440), Poultry (990), Seafood (1100), Seasonings (5968), Vegetables (4048)

Ingredients: Allspice (128), Almond (269), Apple (265), Bacon (180), Baking Powder (399), Baking Soda (294), Basil (347), Bay (281), Bay Leaf (251), Brandy (173), Bread (385), Broth (991), Butter (2236), Cake (140), Capers (120), Carrot (380), Celery (275), Cheese (1290), Cherry (147), Chicken (944), Chilli (427), Chive (148), Cilantro (436), Clove (1486), Cocoa (120), ...

Name: Almond (94), Asparagus (61), Bacon (80), Basil (77), Beans (92), Bell (103), Black (81), Bread (149), Cake (244), Caramel (75), Cheese (358), Cheesecake (65), Cherry (90), Chicken (426), Chili (93), Chocolate (396), Coconut (61), Compote (71), Cookies (78), Corn (138), Cranberry (90), Cream (374), Creamy (60), Crust (74), Dill (67), ...

Recipe Created: April (580), August (456), December (624), February (414), January (319), July (480), June (517), March (566), May (551), November (705), October (507), September (468)

Season: Christmas (227), Easter (54), Fall (1690), Fourth Of July (18), Hanukkah (22), New Year's Day (13), Picnics (81), Spring (1719), St. Valentine's Day (42), Summer (1471), Superbowl (88), Thanksgiving (364), Winter (1358)

Course: Appetizers (615), Bread (233), Breakfast (202), Brunch (200), Condiments (259), Cookies (160), Desserts (1679), Hors D'Oeuvres (197), Main Dish (2157), Salads (560), Sandwiches (123), Sauces (207), Soup (378), Side Dish (757), Snacks (72)

Figure 2: Magnet's interface on a large collection.

Below the query constraints in Figure 1 are the navigation recommendations shown through a set of advisors. The *Similar Items* advisor in the upper left pane suggests additional items (recipes) that have the same *Overall* content (textual and structural) or share a common *Property* with the collection in the main window. The *Refine Collections* advisor shown next (in the middle left) suggests refining the search by one of the metadata attribute axes, as well as by words in the body or in the title of the document. The *Modify* advisor in the lower left allows the user to go to related collections, and in the Figure suggests that the user negate a constraint. Similarly, the *Refinement History* advisor in the navigation pane allows the user to undo previous refinements. Since the user has navigated to a *collection*, the suggestions presented in the navigation pane are those relevant to refining and finding related collections. When individual items are displayed, Magnet suggests collections of items similar to the given one. The user, therefore, can fluidly navigate from items to relevant collections and back as their understanding of their search problem changes.

Advisors thus allow a user to refine a collection of recipes to only show appetizers, by presenting the user with an option to add a constraint limiting the collection. As in the case of the previous example, a user searching for books *by* James, need only enter the query term "James" to first get a list of books containing the word "James," and then add the limiting condition to filter for books *by* James.

When presenting the navigation suggestions to the user, the interface groups suggestions by properties (such as cooking method, ingredients, etc.) and displays the first few values to give the user with appropriate context. Users wanting more choices for a given refinement can ask the user interface to present them with more options (by clicking on the '...').

3.3 Supporting power users

The navigation pane supports power users by allowing them to browse and select multiple navigation suggestions. The context menu on the query allows users to select a compound navigation option like conjunction or disjunction to be applied as a refinement to the current collection. Users can drag suggestions into this compound refinement option, and use them to build a complex query. For example, a user being given Figure 1 can decide that he wants only those items in the current collection that either have a dairy product or a vegetable in them. The user will need to use the context menu to tell the system to build an 'or' refinement, and then drag 'dairy' and 'vegetables' from the panel into the refinement for execution.

Additionally, since the navigation suggestions are created by the user interface inside one or more *collections*, users can navigate to these collections of suggestions (using the context menu) and browse them to find refinements useful for the original query. When users navigate to such collections, the interface provides a subpane within the navigation pane for the user to drag refinement options, and then click on an 'apply' button. Thus, when looking at a collection of recipes, users can navigate to the collection of ingredients, refine the given collection to get those ingredients found only in North America, and then apply the query to either get recipes having an (using or) ingredient found in North America, or to get recipes having all (using and) their ingredients found in North America.

4. THE NAVIGATION ENGINE

Researchers in the HCI community argue that for users to navigate through a given space, with minimal prior knowledge of its organization, each view must present information to help the user make their next navigational decision [16]. They agree that designers need to take into account the learning done by the user during the process and account for the revisions in the search query based on the learned information. Bates [1, 3] shows that when automation is introduced, users want to be able to direct the search. She investigates the level of automation that designers need to give their search system, and recommends support for both the overall search strategies and individual steps such as adding query terms, considering the outcomes of refinement steps, and broadening queries. Magnet aims to support the search process and overall strategies by implementing recommenders of single-step refinement tactics.

The single-step refinement tactics suggested by Magnet's navigation engine are presented via advisors (in the user/search domain) fed by one or more analysts (in the programmatic domain). While analysts represent algorithmic units, they associate their suggestions with advisors, which are targeted towards supporting the users' search strategies.

4.1 Navigation Advisors

Navigation recommendations are posted by analysts on a shared blackboard that is published on the interface by navigation *Advisors*. Each advisor presents a particular type of navigation step. These advisors are integrated in an easily extensible manner to allow schema experts to support new search activities. They are designed to work on the currently viewed item, i.e., they work with both documents and collections (of documents). Applying

Bates' [1] recommendations for the support of single-step refinement tactics, the following advisors have been implemented:

Related Items Suggests options for navigating from the viewed item to a collection of similar items:

- **Sharing a property** That have a given metadata attribute and value in common with the currently viewed item.
- **Similar by Content (Overall)** That share similar content with the currently viewed item. Similar content refers to a fuzzy approach (as determined by a standard learning algorithm) to showing other items having both similar structural elements (properties) and similar textual elements. There are typically two different analysts that are associated with this advisor, one for working with single items and providing other related items, and the other for working with collections and providing more items similar to the items in the collection.
- **Similar by Visit** That were visited the last time the user left the currently viewed item (or left a recently viewed item). This can be thought of as an intelligent history that presents those suggestions that the user has followed often in the past from the current document.
- **Contrary Constraints** That have one of the current collection constraints inverted. This advisor helps users get an overview of other related information that is available.

Refine Collections Suggest navigation based on identified properties and values common to some but not all items in the collection. The selected property and value may be used to either filter the current collection, or remove matching items from the current collection. Alternatively, a user can also use the refinement suggestions as terms to expand the collection and include other matching items.

History Suggests navigation to previously seen items:

- **Previous** That have been seen most recently.
- **Refinement** That are in the refinement trail.

Since there are many possible navigation suggestions to present to the user the navigation advisors are responsible for selecting the most relevant ones and presenting them to the user. The advisors use the analyst-provided information retrieval weights associated with each suggestion to select the navigation suggestions. Analysts providing suggestions to a shared advisor therefore need to have a common approach to giving weights to suggestions. Selected suggestions are presented in the interface typically sorted in an alphabetical order to enable users to search for a particular suggestion.

4.2 The Query Engine

The query engine provides support for resolving the various set concepts. The query engine lets users take the various navigation suggestions (which are predicates) and combine them. By default combination is predicates is by conjunction (and) but the user can also use the context menu to specify disjunction and negations. Furthermore, the query engine provides an extension mechanism for analysts, such that the engine can provide a uniform interface to query both metadata (requiring exact matches) and other attribute value types. For example, the query engine has been extended to uniformly query an external index to support text in documents. Another extension is provided to support numeric attributes in queries by allowing range comparison via greater than and less than predicates.

4.3 Analysts and Blackboard System

Magnet uses a blackboard model [20] to create and suggest navigation options to the user. *Analysts* are triggered by the framework based on the currently viewed (document, collection of documents / result set, query, etc.), and suggest a particular kind of navigation refinement by writing it on the blackboard. The Magnet framework collects the recommendations from the blackboard and presents them with the associated navigation *advisors* to the user.

For example, when the user is viewing a collection of items, a number of analysts are triggered, each providing information to the 'Refine Collection' advisor. One analyst looks for commonly occurring property values and adds them as possible constraints to the current query. Other analysts provide support for keyword search within the collection (as shown under 'Query' in the Navigation Pane in Figure 1) and others provide support for refining the collection based on the type of the data in the collection (for example having range widgets for refining continuous valued types as shown later).

Analysts are triggered by one of many mechanisms. They can be triggered when a user navigates to items of a given type (for example collections or e-mails), and can be triggered by results from other analysts. Once triggered, an analyst can provide a variety of types of recommendations. Most analysts recommend a specific document or collection, others recommend possible query terms to be used in conjunction with the current query (or for a brand new query), and at the most general some analysts specify arbitrary action to be performed upon selection of the suggestion (like running a learning algorithm to find similar documents to the current document).

5. A VECTOR SPACE MODEL

To provide effective navigation suggestions, we fit semistructured data into a vector space model of the form frequently used for textual document retrieval [13]. This allows support for fuzzy notions of similarity on top of the black-and-white boolean queries that typify database search tools. Additionally, it lets us take advantage of the large body of work on query refinement in text repositories [13, 27] can be taken advantage of for designing navigation advisors. We describe an approach to adapting the basic concepts from the field and show how we apply them.

The vector space model maps text documents to vectors, with a coordinate for each word in the corpus, whose value is set according to the number of occurrences of that word in the document. While the approach is limited in that the two sentences 'John ran faster than David' and 'David ran faster than John' lead to the identical representations, the model is still helpful in that it can say that John and David where involved in running and that one of them is faster. Improvements to the model include removing frequently occurring words (stop-words), removing common suffixes (stemming) and normalizing tenses of words. As an example, we might map "Betty bought some butter, but the butter was bitter" to give the vector:
$\langle betty \Rightarrow 1, buy \Rightarrow 1, some \Rightarrow 1, butter \Rightarrow 2, bitter \Rightarrow 1\rangle$

5.1 Building the model

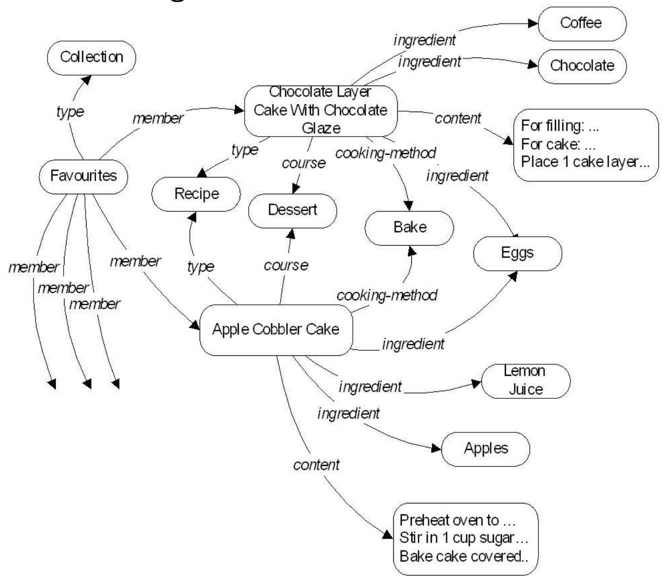


Figure 3: An RDF Graph of some recipes.

Document ID	type: Recipe	course: Main dish	course: Dessert	course: Salad	cooking-method: Bake	cooking-method: Roast	ingredient: Coffee	ingredient: Eggs	ingredient: Lemon Juice	ingredient: Apples	ingredient: Turkey	title: apple	title: vinegar	title: cobbler	title: cake	title: chocolate	title: cider	content: filling	content: cake	content: layer	content: ...	⋮
Choc-Layer-Cake	1	0	1	0	1	0	1	1	0	0	0	0	0	0	1	2	0	1	9	3	...	...
Apple-Cobb-Cake	1	0	1	0	1	0	0	1	1	1	0	1	0	1	1	0	0	2	6	0	...	...
Fennel-Apple-Salad	1	0	0	1	0	0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	...	...
Cider-Basted-Turkey	1	1	0	0	0	1	0	0	0	1	1	1	1	0	1	0	1	5	0	0	...	...
Apple-Crunch-Pie	1	0	1	0	1	0	0	0	0	1	0	1	0	0	1	0	0	1	0	0	...	...

Figure 4: The vector space model representation of the recipes (corresponding to Figure 3). The values in upper case (such as for the type, course, cooking method, and ingredient attributes) are objects, while the values in lower-case (such as the title and content attributes) are text strings which have been further split-up.

We apply the vector space model to semistructured. For a given item, for example 'Apple Cobbler Cake' shown in Figure 3, each attribute/value pair associated with a given piece of information is treated as a coordinate in a vector space model (as shown in Figure 4)—just as terms usually are for text documents. As in the traditional vector space model individual words in paragraphs of text are split up and represented as coordinates. When attribute values are other items (rather than primitive elements like numbers or strings), we represent them by unique identifiers in the model. In other words, we have one coordinate for each attribute-value pair.

With the vector space model, a common extension calls for having multiple word phrases as coordinates. While this form of extension is also helpful in semistructured version of the model, another axis of extension is also helpful with semistructured data—that of composing multiple attributes and thereby providing support for "transitive" relations in the model. For example, since documents have authors and authors can have fields of expertise, a property that might be helpful for browsing is "the author's field of expertise" and it might therefore be helpful to encode such a relationship in the model. Ideally the model should be built and provide results based on all possible attribute compositions. However, semistructured information can be highly interconnected, and therefore for performance reasons Magnet only selectively adds attribute compositions into the model. The attribute compositions that should be included as coordinates in the vector space model are listed as schema annotations in the data store. These annotations can be entered by schema experts or by advanced users through the interface (in the context menu). Additionally, just as systems can be built to learn phrases for use in traditional vector space models, we expect that systems might ultimately learn to automatically detect and incorporate important compositional relations.

5.2 Implementation considerations

Once the vector space model is extended for semistructured data, a few transformations are applied to ease integration with traditional information retrieval techniques. For example, in Magnet performance is enhanced by "indexing" the data in advance (as it arrives)—an appropriate vector is built for each item, and stored in a vector-space database (the Lucene text search engine [15] is used for this purpose).

The semistructured vector space model can then apply those *normalizations* traditionally recognized as being effective in the vector space model [24]: tf.idf term weighting which divides the term-frequency of an attribute/value coordinate by the (log of the) number of times that attribute/value appears in the corpus, helps the system ignore those attribute values that are very common. We normalize each document vector to a length of one, in order to give objects equal importance rather than giving more importance to items with more metadata. New weights of the vector space model are therefore calculated as follows:

$$\text{term-weight} = \log(\text{freq} + 1.0) \times \log \frac{\text{num-docs}}{\text{num-docs-with-term}}$$

$$\text{normalized-weight} = \frac{\text{term-weight}}{\sqrt{\sum_{\forall \text{term}} \text{term-weight}^2}}$$

Semistructured data adds a level of complexity to the normalization approach, since items can differ both in the number of attributes they have and in the number of values they have for a given (multivalued) attribute. While selecting the appropriate form of normalization can be task dependent, we have chosen to use an approach similar to that used in ranking documents in Lucene, i.e., to first divide each term frequency by the number of values for the attributes, and then to normalize each object as mentioned above. This approach gives equal importance to different attributes in a document, i.e. for an email, the importance of the subject is the same as the importance of the body.

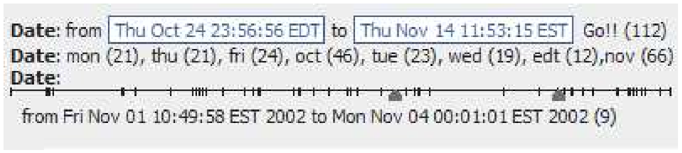
5.3 Applying traditional techniques

Once the semistructured-extended vector space model is normalized, traditional information retrieval techniques can easily be adapted to support it. The *similarity* between two documents or between a document and a query can be determined by a traditional dot-product between the two vectors [12]. The dot-product is used since documents with many terms in common, which intuitively are similar, have a larger dot product. In a natural generalization, we determine the similarity of a document to a collection by dotting the document's vector with an "average member" of the collection using a vector made up of the (normalized) *sum* of the vectors in the collection.

To suggest terms to add to the query we use a *query refinement* technique, which seeks words in textual queries that are common (but not too common) in the current result set [27]. Query refinement can be applied in the semistructured data case to suggest attributes and values be added to the current query to refine the result set. Given that normalization of vectors in the model was done explicitly to decrease weights on terms occurring too frequently, applying this technique involves just picking terms in the average document having the largest normalized term weights.

5.4 Numeric Attributes

The vector space model works well when the interesting thing about attribute values is whether they are equal. However, it is common to encounter attributes such as dates that are numeric; in this case, it is not just equality but also numerical closeness that can indicate similarity. Here the interface makes it possible to allow the user to refine a collection by specifying a specific range for this property. The range selection controls shown in Figure 5 applies to a collection of e-mails exposed in the user interface and uses two sliders to select the upper and lower boundary presenting hatch marks to represent documents thus showing a form of query preview.



Date: from Thu Oct 24 23:56:56 EDT to Thu Nov 14 11:53:15 EST Go!! (112)
Date: mon (21), thu (21), fri (24), oct (46), tue (23), wed (19), edt (12),nov (66)
Date:
from Fri Nov 01 10:49:58 EST 2002 to Mon Nov 04 00:01:01 EST 2002 (9)

Figure 5: Possible interfaces that can be used to show date options

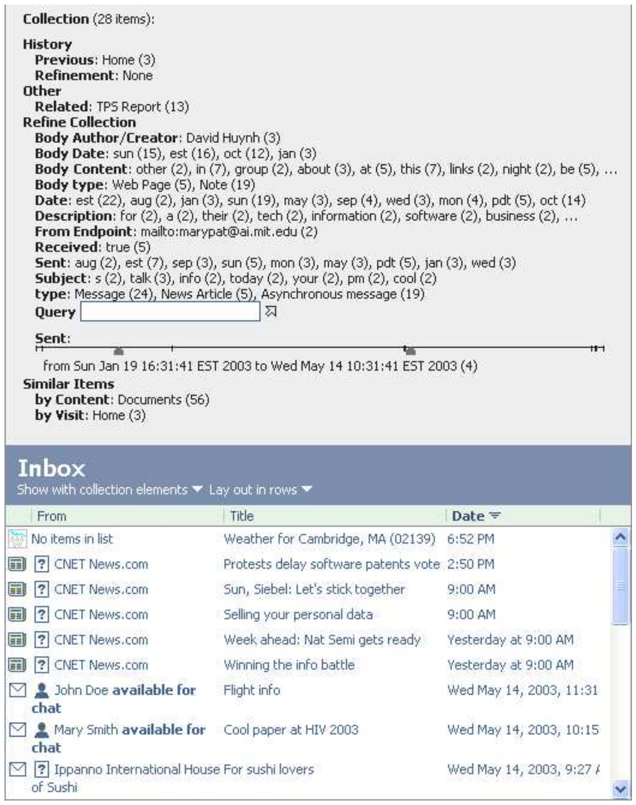
We support these continuous-valued attributes by extending the query engine to use attribute types and use the information to provide support for range comparison operations in queries. Furthermore, we support algorithms for measuring similarity by *also* converting these attribute values into numbers, thereby allowing two e-mails received a day apart: 'Thu July 31, 2003' and 'Fri August 1, 2003' to have some similar attributes (rather than just having the year be common in them). To keep the numeric values (which might be arbitrarily large) from swamping other coordinates in the vector space model when we normalize, we map the numeric range to the first quadrant of the unit circle, so that all values have the same norm but different values have small dot product.

6. EVALUATION

Magnet was designed to provide a usable yet flexible way to meet users search needs with semistructured data. Towards this goal, we conducted preliminary evaluations on three axes. We first looked at Magnet's flexibility to work with varying data source. We then evaluated browsing flexibility. Finally, we tested the interface through a user study.

Our ability to conduct a user study was limited by the challenge of developing good evaluation metrics for browsing, and in particular by the absence of a preexisting semistructured corpus with questions and predetermined answers we could compare it. Thus, much of the study is qualitative in nature. Nonetheless, various interesting points were revealed.

6.1 Datasets



Collection (28 items):

History
Previous: Home (3)
Refinement: None
Other
Related: TPS Report (13)
Refine Collection
Body Author/Creator: David Huynh (3)
Body Date: sun (15), est (16), oct (12), jan (3)
Body Content: other (2), in (7), group (2), about (3), at (5), this (7), links (2), night (2), be (5), ...
Body type: Web Page (5), Note (19)
Date: est (22), aug (2), jan (3), sun (19), may (3), sep (4), wed (3), mon (4), pdt (5), oct (14)
Description: for (2), a (2), their (2), tech (2), information (2), software (2), business (2), ...
From Endpoint: mailto:marypat@ai.mit.edu (2)
Received: true (5)
Sent: aug (2), est (7), sep (3), sun (5), mon (3), may (3), pdt (5), jan (3), wed (3)
Subject: s (2), talk (3), info (2), today (2), your (2), pm (2), cool (2)
type: Message (24), News Article (5), Asynchronous message (19)
Query

Sent:
from Sun Jan 19 16:31:41 EST 2003 to Wed May 14 10:31:41 EST 2003 (4)
Similar Items
by Content: Documents (56)
by Visit: Home (3)

Inbox
Show with collection elements ▾ Lay out in rows ▾

From	Title	Date
No items in list	Weather for Cambridge, MA (02139)	6:52 PM
CNET News.com	Protests delay software patents vote	2:50 PM
CNET News.com	Sun, Siebel: Let's stick together	9:00 AM
CNET News.com	Selling your personal data	9:00 AM
CNET News.com	Week ahead: Nat Semi gets ready	Yesterday at 9:00 AM
CNET News.com	Winning the info battle	Yesterday at 9:00 AM
John Doe **available for chat**	Flight info	Wed May 14, 2003, 11:31
Mary Smith **available for chat**	Cool paper at HIV 2003	Wed May 14, 2003, 10:15
Ippanno International House of Sushi	For sushi lovers	Wed May 14, 2003, 9:27 /

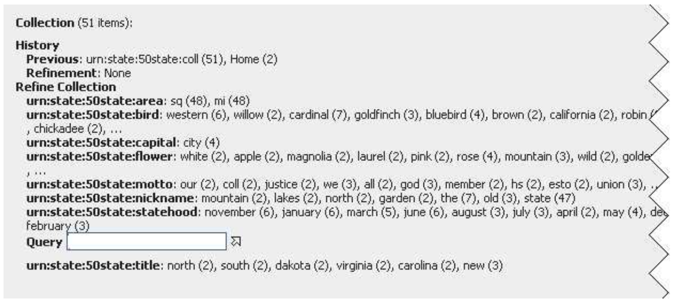
Figure 6: A view of the output provided by the navigation system in the user's Inbox.

We tested data generated by the underlying system and then tested on data from external sources. We used the system on a collection of e-mails in the system's Inbox (Figure 6). Magnet suggested refining by the document type since the inbox contains messages as well as news items from subscription services. The system also used the annotation that body is an important property to compose with a second level of attributes and suggested refining by the type, content, creator and date on the body (as can be seen in the figure). Additionally, the system provided a range control to refine by the sent dates of items in the inbox, and gave the user the option of querying within the collection.

The system was tested on four external datasets. Two of these were a collection of information about 50 states provided as a comma separated file and an RDF version of the CIA World Factbook. We expected only limited results in both datasets since they have document properties encoded as human-readable strings rather than marked up semantically.

For the 50 states dataset[1], shown in Figure 7, RDF identifiers were displayed since the dataset did not have human-readable labels associated with them. This system did point out interesting attributes that a searcher might need, for example, the fact that seven states have 'cardinal' in their bird names, and allowed for clicking on the name to give the a collection of states with the property. Adding labels on each property and annotating the area property to indicate that it is an integer made the interface more user-friendly (as

[1]Extracted from `http://www.50states.com` and made available as a comma-seperated values file.

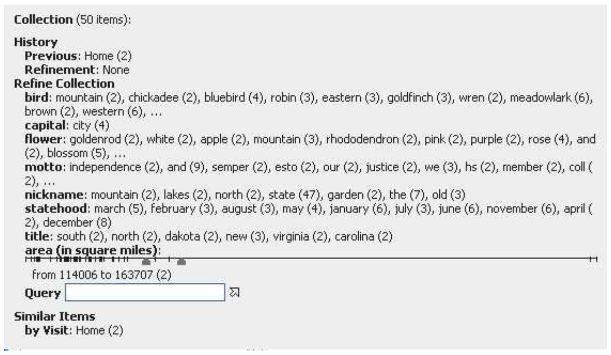


Collection (51 items):

History
Previous: urn:state:50state:coll (51), Home (2)
Refinement: None
Refine Collection
urn:state:50state:area: sq (48), mi (48)
urn:state:50state:bird: western (6), willow (2), cardinal (7), goldfinch (3), bluebird (4), brown (2), california (2), robin
, chickadee (2), ...
urn:state:50state:capital: city (4)
urn:state:50state:flower: white (2), apple (2), magnolia (2), laurel (2), pink (2), rose (4), mountain (3), wild (2), golde
, ...
urn:state:50state:motto: our (2), coll (2), justice (2), we (3), all (2), god (3), member (2), hs (2), esto (2), union (3), .
urn:state:50state:nickname: mountain (2), lakes (2), north (2), garden (2), the (7), old (3), state (47)
urn:state:50state:statehood: november (6), january (6), march (5), june (6), august (3), july (3), april (2), may (4), de
february (3)
Query
urn:state:50state:title: north (2), south (2), dakota (2), virginia (2), carolina (2), new (3)

Figure 7: The output provided by Magnet's interface for the 50 states dataset (as given).

shown in Figure 8) by showing expected label and range controls for the area. The figure clearly shows one state (Alaska) having a much larger area than the rest, and shows that a number of states have the same bird and flower. Similarly, the CIA World Factbook[2] results with Magnet improved with label and attribute-value type annotation. The navigation system did recommended navigating to countries that have the same independence day or currencies.



Collection (50 items):

History
Previous: Home (2)
Refinement: None
Refine Collection
bird: mountain (2), chickadee (2), bluebird (4), robin (3), eastern (3), goldfinch (3), wren (2), meadowlark (6), brown (2), western (6), ...
capital: city (4)
flower: goldenrod (2), white (2), apple (2), mountain (3), rhododendron (2), pink (2), purple (2), rose (4), and (2), blossom (5), ...
motto: independence (2), and (9), semper (2), esto (2), our (2), justice (2), we (3), hs (2), member (2), coll (2), ...
nickname: mountain (2), lakes (2), north (2), state (47), garden (2), the (7), old (3)
statehood: march (5), february (3), august (3), may (4), january (6), july (3), june (6), november (6), april (2), december (8)
title: south (2), north (2), dakota (2), new (3), virginia (2), carolina (2)
area (in square miles):
from 114006 to 163707 (2)
Query

Similar Items
by Visit: Home (2)

Figure 8: Adding annotations for state area and labels for properties.

Two other datasets used for evaluating the system were independent external conversions to RDF of the data behind MIT OpenCourseWare[3] and ArtSTOR[4] These datasets did have label and attribute-value annotations, allowing Magnet to present easy to understand navigation suggestions. The attributes that Magnet suggested for refinement did include options that where not human-readable. While these options were determined to be algorithmically significant for refining, they were not deemed important for end-user navigation. While Magnet does provide custom annotations to hide such attributes, it does indicate room for improvement in the actual algorithm for selection of term weights.

6.2 Browsing Flexibility

We tested the flexibility of the navigation engine using the collection of topics provided by the INitiative for the Evaluation of XML retrieval (INEX) [9] (created by the DELOS Network of Excellence for Digital Libraries and the IEEE Computer Society). The collection is used for evaluation of search result rankings and consists of two types of search topics: those having text only (words in document) and topics that have both textual information and structural information. While Magnet does not currently support ranking documents, these search topics were used to evaluate the flexibility of the engine.

The first set of search topics, that Magnet's flexibility was examined against where those containing both text and structure. One such query required the "Vitae of graduate students researching Information Retrieval" from the given corpus. Magnet's navigation engine did have the flexibility to retrieve most of the documents needed. However, the system would have presented a better interface if annotation had been provided for the composition of the relations, i.e., the multiple steps used in the XML documents. Since Magnet has been designed to support the more general graphs of data (which can have cycles), as opposed to trees found in XML (having finite depths), Magnet would not follow multiple steps by default. Telling Magnet that the information is structured as a tree, or using the set of possible XML paths as indication of possible compositional relationships would have provided a cleaner interface. A second limitation of Magnet when compared to the INEX search topics was that Magnet's focus of providing a browsing interface provides a limited exposure to the query engine by default. Magnet would have been able to better support INEX searches with integrated support for a larger set of basic queries, for example, a user might want to look at all recipes having 5 or fewer ingredients.

The other set of topics provided by INEX were topics consisting only of text portions, for example: "software cost estimation". Such searches involved the direct application of traditional IR techniques to find the relevant text. Since Magnet is built on these techniques, it would have been able to retrieve all such documents. In such situations, the only weakness with Magnet compared to other systems was the absence of document reordering, for example, as shown by Kamps et al. [17] biasing results to favor large documents can improve such queries since the results are otherwise swamped by significant numbers of small documents. Such improved results can be directly extended to Magnet.

6.3 User Study

A preliminary user study was conducted to understand issues raised by the system, and to ensure that the user was not swamped by too much advice. Our interface was modeled after Flamenco's faceted metadata navigation system [28], and we therefore built a baseline system consisting of navigation advisors suggesting refinements roughly the same as those in the Flamenco system. The baseline system also included terms from the text of the documents and allowed users to negate the terms by right clicking on them. The second interface represented the complete system, and had all the advisors mentioned earlier. In particular, beyond collection refinement, it allowed users to get documents similar to the current document by analyzing the document content and explicitly suggested the user with contrary options (negating a current search constraint).

Prior to the study, feedback from 6 users was used to iteratively improve the interface and resolve performance issues. The study used data from 6,444 recipes and metadata extracted from the site Epicurious.com. 244 ingredients were semi-automatically extracted from the recipes and grouped to supplement the data (as shown in Figures 1 and 2). The study included two undirected tasks (the first and last tasks) where the users had minimal constraints, and were asked to simply 'search recipes of interest'. Two directed tasks were also

[2]Available in RDF at `http://www.ontoknowledge.org/oil/case-studies`.
[3]Available at `http://ocw.mit.edu/`
[4]A non-profit organization to develop and distribute electronic digital images. `http://www.artstor.org/`.

given to users where they were constrained to find particular kinds of recipes:

- When an aunt left your place, you found a recipe that she had been excited about and had left behind by mistake. While looking at the recipe you realize that it has walnuts and that your uncle is allergic to nuts. Find the recipe on the system and a few 2-3 other related recipes that your uncle and aunt may like.
- You are planning a party for some friends. The party is supposed to be a Mexican themed night, and you have been asked to plan a menu. Make sure you have some soups or appetizers, as well as salads and desserts on top of the meal. Try to include some of your favorite ingredients that you mentioned earlier.

6.3.1 Results and Discussion

The study was advertised via paper posters through the Computer Science building. There were 18 participants is the study, all of whom were graduate students in our program. The study brought out a number of qualitative differences between the two systems. Only one user complained that the navigation options were overwhelming and this was on the baseline system. Since there were minor differences between the baseline system's interface and the Flamenco user interface, these differences may be worth investigating.

Most mistakes while doing the tasks seemed to be due to capture errors [21], i.e. users performed an incorrect but more easily available sequence. For example, in the first directed task, users were expected to find recipes similar to a target recipe but that did not have nuts in them. Some users attempted to find recipes by adding 2 or 3 ingredients, *including* nuts, as constraints to get a list of recipes, and then issuing a refinement to exclude items with nuts, producing the empty result set. In other cases, as well users seemed to be mapping negation to 'find similar but not'. The Magnet interface currently presents users with navigational suggestions that are either explicitly boolean (adding refinement constraints) or explicitly fuzzy (getting similar documents), however, since users find it difficult to work with zero results [8], it may be worth modifying the queries to perform more fuzzily in the case when zero results would have been returned otherwise.

Users did seem to be able to use the navigation options when they created the sub-problems themselves. For example, in the second directed task, we expected that users would look at the subset of Mexican dishes and then explore within them. However, some users searched for one or two of their favorite ingredients, then refined by Mexican cuisine. Another user searched for her favorite dish first, asked the system to give similar recipes and then refined by Mexican. Users seemed to not have problems using the extra features (over the baseline systems) either when they were doing an undirected part of the task, or after they used it once or twice.

Even though users' errors seemed to indicate that they found the system complex and needed time to get used to it, users were more successful using the complete system than using the base system. For example, most users on both systems had a hard time getting negation right as needed in the first directed task (removing nuts from the listed recipes). However, even when not sure how to proceed with the system, users working with the complete system found it easier to accomplish those tasks since the contrary advisor would suggest negation to get them started in the process. Users also found more recipes to match the criteria of the tasks. For the first directed task, users found on average 2.70 recipes with the complete system and 1.71 recipes with the baseline system; and for the second task, users found on average 5.80 recipes with the complete system and 4.87 recipes with the baseline system. Since the study was small, we cannot claim statistical significance in the data.

7. CONCLUSIONS AND FUTURE WORK

As rich semistructured data under numerous and often-violated schemata becomes more pervasive, we must deploy tools that help unsophisticated end users locate information. The orienteering paradigm, in which users browse large collections and "home in" on information of value, is an effective tool for such users [25]. We have presented a tool that aims to support users browsing in large collections of semistructured data. Without requiring schema-specific customization, Magnet presents users with useful query refinement and navigation steps that they can use to seek out information of interest. Magnet demonstrates that some of fuzzy information retrieval techniques that are effective in textual information retrieval can be applied effectively to semistructured information.

We have conducted preliminary evaluations of the approach on different data sources (with varying schema information), examined the flexibility of the browsing interface on queries from XML retrieval initiatives, and conducted a user-study with a system shown to work with metadata browsing. Evaluations indicated a number of directions for building on the system. When dealing with datasets, it was found that a number of simple annotations are often needed such as indicating attribute value types or attribute compositions. Heuristic rules or learning approaches to determine such annotations will be helpful. Additionally, even though users in the user-study had fewer problems and found more items using the tool when compared to a base system, a few interface issues were found. The evaluation indicated a need to examine finer interface details like the ease of negating and the learning curve, before conducting a larger scale evaluation of the system.

8. ACKNOWLEDGEMENTS

We would like to thank David Huynh for comments and help with this research, Dennis Quan for developing most of the system that Magnet runs on, and Prof. Samuel Madden, Prof. Randall Davis, and Suelene Chu for help with the content of this paper. This work was supported by the MIT Simile project, the MIT-NTT collaboration, the MIT Oxygen project and a Packard Foundation fellowship.

9. REFERENCES

[1] Marcia J. Bates. Information search tactics. *Journal of the American Society for Information Science*, 30(4):205–214, 1979.

[2] Marcia J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Review*, 13(5):407–424, October 1989.

[3] Marcia J. Bates. Where should the person stop and the information search interface start? *Information Processing and Management*, 26(5):575–591, 1990.

[4] David Carmel, Yoelle S. Maarek, Matan Mandelbrod, Yosi Mass, and Aya Soffer. Searching XML documents via XML fragments. In *Proceedings of the 26th annual international*

ACM SIGIR conference on Research and development in informaion retrieval, pages 151–158. ACM Press, 2003.

[5] Tiziana Catarci, Maria Francesca Costabile, Stefano Levialdi, and Carlo Batini. Visual query systems for databases: A survey. *Journal of Visual Languages and Computing*, 8(2):215–260, 1997.

[6] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/Gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR*, pages 318–329. ACM Press, 1992.

[7] Daniel Egnor and Robert Lord. Structured information retrieval using XML. In *Working Notes of the ACM SIGIR Workshop on XML and Information Retrieval*, 2000.

[8] User Interface Engineering. Users don't learn to search better. `http://www.uie.com/Articles/not_learn_search.htm`.

[9] Norbert Fuhr, Mounia Lalmas, and Saadia Malik, editors. *INitiative for the Evaluation of XML Retrieval (INEX). Proceedings of the Second INEX Workshop*, December 2003.

[10] George W. Furnas and Samuel J. Rauch. Considerations for information environments and the NaviQue workspace. In *Proceedings of the third ACM conference on Digital libraries*, pages 79–88. ACM Press, 1998.

[11] Roy Goldman and Jennifer Widom. Interactive query and search in semistructured databases. In *Proceedings of the First International Workshop on the Web and Databases (WebDB '98), Lecture Notes in Computer Science 1590*, pages 52–62. Springer-Verlag, March 1998.

[12] Donna Harman. Ranking algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information retrieval: data structures and algorithms*, chapter 14, pages 363–392. Prentice-Hall, Inc., 1992.

[13] Donna Harman. Relevance feedback and other query modification techniques. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information retrieval: data structures and algorithms*, chapter 11, pages 241–263. Prentice-Hall, Inc., 1992.

[14] Haystack. The universal information client. `http://haystack.lcs.mit.edu/`.

[15] The Apache Software Foundation Jakatra Project. The Lucene search engine. `http://www.lucene.com/`.

[16] Susanne Jul and George W. Furnas. Navigation in electronic worlds: Workshop report. *SIGCHI Bulletin*, 29(4):44–49, October 1997.

[17] Jaap Kamps, Maarten Marx, Maarten de Rijke, and Börkur Sigurbjörnsson. XML retrieval: What to retrieve? In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 409–410. ACM Press, 2003.

[18] O. Lassila and R. Swick. Resource description framework (RDF): Model and syntax specification. `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222`, February 1999. W3C Recommendation.

[19] Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass, and Jennifer Widom. Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3):54–66, 1997.

[20] H. Penny Nii. The blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine*, 7(2):38–53, Summer 1986.

[21] Donald A. Norman. Design rules based on analyses of human error. *Communications of the ACM*, 26(4):254–258, 1983.

[22] Peter Pirolli and Stuart Card. Information foraging in information access environments. In *Conference proceedings on Human factors in computing systems*, pages 51–58. ACM Press/Addison-Wesley Publishing Co., 1995.

[23] The Simile Project. Longwell suit of web-based RDF browsers. `http://simile.mit.edu/longwell/`.

[24] Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, and David R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[25] Jaime Teevan, Christine Alvarado, Mark S. Ackerman, and David R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 415–422. ACM Press, 2004.

[26] Agathoniki Trigoni. Interactive query formulation in semistructured databases. In *Proceedings of the Fifth International Conference on Flexible Query Answering Systems (FQAS 2002), Lecture Notes in Computer Science 2522*, pages 356–369. Springer-Verlag Heidelberg, October 2002.

[27] Bienvenido Vlez, Ron Weiss, Mark A. Sheldon, and David K. Gifford. Fast and effective query refinement. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 6–15. ACM Press, July 1997.

[28] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408. ACM Press, 2003.

[29] Franois Yergeau, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. Extensible markup language (XML). `http://www.w3.org/TR/2004/REC-xml-20040204/`, February 2004. W3C Recommendation.