TOOLS ENGINEERED SPECIFICALLY FOR INFORMATION SPACE ORGANIZATION

by

STEPHEN DAVIES
B.S. Rice University, 1992
M.S. University of Colorado, 1996

A proposal submitted to the
University of Colorado Department of Computer Science
in partial fulfillment of
the PhD Comprehensive Examination requirement

April 21, 2004

Committee members: Roger King (chair), Ken Anderson,
Greg Grudic, Clayton Lewis, Jim Martin

TABLE OF CONTENTS

1. Introduction and motivation

The burgeoning "information overload" problem plaguing today's technology users is well-documented. On nearly every front, the amount of data collected electronically far outstrips our ability to keep pace with understanding it. We have embedded devices with sensors that continually gather data; online digital libraries that regularly add new material as well as old; supermarket chains that capture and archive our every purchase; and of course the World Wide Web, which by conservative estimates is growing at a rate of over a hundred pages per minute[78]. It's no secret that by themselves, these vast megabytes mean nothing. In general, only the interpretation and deep analysis by human beings gives rise to productive knowledge that can be of practical benefit. Anything less is a waste of time.

The tools currently available to assist in this quest for knowledge are principally two: query languages, in their various forms, and the discipline of data mining. The former is well-known to virtually any consumer of data – users (or programs) pose a series of concrete inquiries to some kind of interface, which attempts to extract bits of knowledge from an enormous data store. Sometimes a query is an intricate piece of code, such as an SQL statement, that navigates and pulls together a series of tables and values to arrive at a precise answer. In other cases, for instance a Google web search, it's more of a shot in the dark: a group of terms suggests the desired content which the search engine aims to find. In either case, between the human and the information lies a narrow channel through which questions are posed and results given. Researchers in the fields of query optimization and information retrieval have invested tremendous effort in improving this channel, in order to make the results speedier and more satisfactory.

Data mining takes a different approach; the emphasis is less on ad hoc interaction and more on the discovery of patterns deeply embedded in the data which can only be uncovered by exhaustive statistical analysis. In some cases, human observers bring a great deal of structure to the data at the outset: they know the relevant categories to which individual data records belong, and seek complex rules by which future records can be automatically classified. Or they may approach a large data set with a particular subset of variables in mind, and want to verify mathematical correlations between them. In still

other, "unsupervised" contexts, investigators stay as unbiased as possible, and seek to discover in what ways the data naturally groups itself. What all these scenarios have in common is that they usually involve an offline, time-consuming process, and that the investigators include experts in both the relevant domain and statistical methods. It is assumed that the time taken to study every facet of the data set and the patience required to coax the maximally revealing conclusions from it is well worthwhile.

These two approaches have filled important needs. Years ago, the speed and expressiveness of database queries made it profitable to collect the volumes of tabular data in the first place. More recently, the applications of data mining have impacted innumerable fields, leading both to strategic-level discoveries within populations and to the possibility of automation in areas where none before existed. And it is difficult even to imagine tapping any portion of the Web's vast information store without the power and simplicity of a search engine interface. Clearly, these techniques have settled into important niches, and improvements along the same general lines of attack are well justified.

Yet I believe a need is present – and actually a rather common one – that is largely unmet by either of these two approaches. Little attention has been given to it, probably because the aforementioned techniques can be used indirectly to yield a partial solution, with the user filling in the gaps manually. But I know of no application or method that attacks the problem directly. Let me explain what I mean by revisiting the limitations of current technologies.

1.1. The "query-response" model

Consider the query-response model. It's really best for the person who confronts an overwhelming amount of data that they could never possibly fully digest. The idea is to allow this user to relatively painlessly extract some piece of helpful information, so that they can get back to their life (and safely away from the database!) as quickly as possible. It is taken for granted that the vast majority of the available information is irrelevant, and the goal is to filter it on the user's behalf. This is why nearly every query boils down to, "find me some small fraction of the data that matches *this*, and please

don't bother me with anything else." It is also why a stateless interface is so common – and appropriate – for such tools. Each nugget mined is an isolated piece of information, presumably unrelated to other things that may be asked for later.

This is perfectly sufficient in many cases. To use the Web as an example, there are times when I simply want to find a single e-mail address, a download site, a publication, or the score of a baseball game. I'm not attempting to make any connections or form any grand theories about the available information as a whole. A stateless interface is perfect, since it allows me to pose standalone queries and get individual answers quickly.

But there are times when I have a completely different goal in mind, and I end up using Google to accomplish it indirectly. Sometimes I want to really *understand* some part of the information space in its entirety – what kinds of pages are out there on a given topic, and what kinds are missing; roughly how many are of the various kinds, and how they are related to each other. Perhaps I'm interested in buying a new disk drive, but I don't really know much about disk drives – what different kinds there are, how many vendors offer each kind, what the relative prices are. Or maybe I want to get the "feel" for the literature in a certain research area: who the principal researchers are in that field, who works with whom and who's studying what, how the topic is partitioned into different specialties, etc. In these situations I'm seeking much more than definitive answers to a few isolated questions. Instead, I'm building up a mental model as I go – gathering information, relating it to what else I know, making judgments about what is related to what, and gradually forming a complete mental picture of how the information is assembled as a whole.

Google, of course, has no idea that I'm trying to do this. It sees no relationship between my various queries, and has no concept of what I'm really trying to find out. Each query is a disconnected, context-free event that provides some piece of information detached from everything else, and it is up to me to assemble these bits myself into a meaningful mental framework. It's rather like a consultant who is supremely brilliant and utterly dull at the same time: he knows all the answers to every individual question I might ask, but throughout the interview it never dawns on him what I'm really driving at.

He coughs up detail after detail, but sees no relationship between them and can help me see none.

I contend that my goal here is not an uncommon one. There are many situations in which a user has a vested interest not in merely extracting facts from an information space, but in forming a broad and thorough understanding of it. Indeed, this kind of intuition is what gives experts the background to render individual detailed decisions anyway. As a prospective homeowner, I go to a realtor in part because I believe the end result will be a particular property that is well-suited to me. But what gives the realtor the ability to recommend such a house is that they are able to draw on a deep understanding of what kinds of properties are available in the area, which ones might be best suited to me, what issues are involved that I may not have considered, and so on. The final decision is a fact, a detail. But the realtor is in the business of building expertise and intuition about the market as a whole, so that they can render many such decisions. And it is hardly ideal to try and build such a comprehensive understanding by selecting straws from a haystack one at a time.

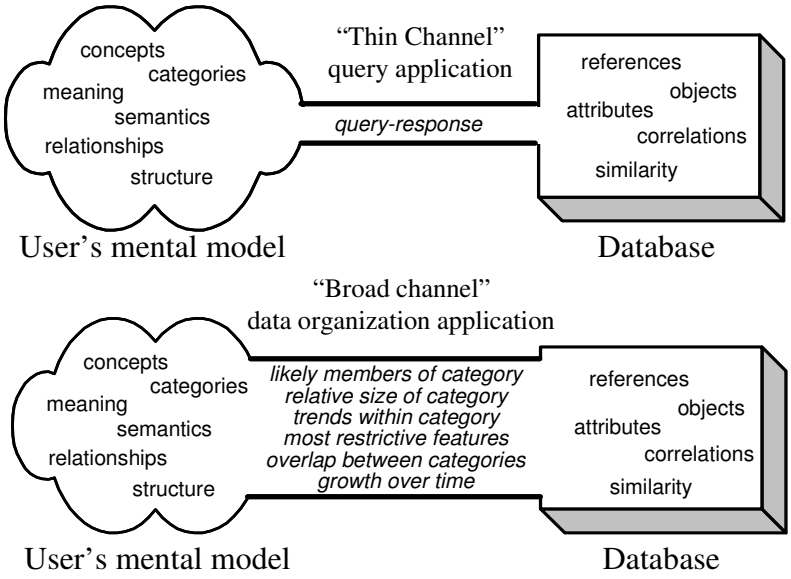

Figure 1. Two different approaches in database interfaces. In the top picture, each database interaction is an isolated question and answer, resulting in a single disconnected fact. The user is forced to infer the broader structure of the information space by manually assembling these facts into a mental framework. This is typical of today's technology. In the bottom picture, the interface accommodates a higher level of abstraction, modeling the user's mental constructs explicitly and allowing questions to be asked directly about them. The structure that the navigation experience reveals, then, is not only in the user's head; it is also embodied in the tool. This second approach is what I intend to investigate in this thesis.

In short, the typical query application does not offer us any explicit way to track or organize or predict or evaluate the emerging conceptual structure that we assemble as we roam. But it could. We could broaden the channel, so to speak, between the user and the information (see Figure 1), so that the higher-level abstractions people typically think in terms of – categories, trends, relationships – are explicitly modeled and can be manipulated. The result of browsing, then, is not simply a transient succession of items that have been viewed, forever lost outside the user's own head. It is rather a tangible artifact, a record of what has been learned and how it can be generalized to other information not yet seen.

1.2. A new kind of data mining

This is precisely where data mining could be leveraged to immense benefit. One of the impediments to the above approach is that the mental constructs users relate to are often abstract and "fuzzy." This defies the traditional database paradigm, which assumes large volumes of concrete, unambiguous elements. It is difficult, using a concrete query language like SQL, to specify anything like a "category of instances" unless it happens to coincide directly with an attribute value. A political strategist can thus ask questions about "registered Democrats" fairly easily. But trying to get a handle on "soccer moms" is much more difficult. Here, the category represents a valid and important phenomenon that has meaning in the real world, but the criteria for inclusion are nebulous and perhaps elusive for humans to define. Enter data mining. With enough individual instances manually identified, machine learning algorithms can learn rules that can generalize to the population as a whole. All that is required is for users to browse from instance to instance – a process already familiar from the query-response paradigm – and inform the system about whether each instance is a member of the category. The large body of standard data mining algorithms can then be applied to help the user interactively recognize patterns as they are encountered.

Notice that this is much different than standard data mining. Standard data mining presupposes large quantities of training examples that have already been labeled, and works offline to produce classifiers for unseen data. But what I propose is an

interactive process where the user approaches the system without much a priori knowledge about what the database contains, without any hypotheses, and without any intuition about what the appropriate "labels" even are. The user browses without realizing that "data mining" is even happening at all – the algorithms operate in the background, helping to identify patterns in response to user feedback.

Equally important is that this process is inherently *subjective*. Standard data mining presupposes objectivity: when presented with a list of hard facts, each one "correctly" labeled, it attempts to learn the "correct," optimal classifier. But in the system I propose each user brings their own conceptions, priorities, goals, and biases to the table. To one user, the data should be naturally organized along certain lines, but to another, a completely different set of relationships is appropriate. Neither is more "correct" than the other, and it is the system's job to learn an individual's personal semantics as the browsing experience progresses. In fact, I would argue that there really is no such thing as "the" semantics of a set of data. Data is just data. Semantics only emerge as a user introduces their own subjective evaluation to it. The system's task is to learn from what the human identifies as semantically related, and to dynamically adjust to this in order to help the human recognize the particular patterns that are important to them.

1.3. A new technique: the "interactive data organization" model

My goal is to design a tool specifically to help a user organize and categorize an information space. People will naturally form mental models of domains they explore. But rather than abandoning the user to carry out that process themselves, providing them with only a sequence of facts, I aim to incorporate their mental constructs explicitly into the user interface experience. I want to broaden the communication channel between user and database, so that the categories of mental perception can be stated, examined, and reasoned about in their own right. In this way, the level of abstraction is raised to the user's more natural cognitive model, and the tool directly assists them in accomplishing their *real* objective.

The particulars of how a user would best interact with such a system will be a major focus of my study. But I have already identified a number of probable characteristics, including:

- **Instance-based**. The most prominent element of the interface is a browser through which users navigate from instance to instance[1]. This is as opposed to viewing data along particular dimensions (e.g., breaking out a given attribute from all applicable instances) or in aggregation (e.g., dealing with an entire "table" of tuples of a given type.) My belief is that users naturally reason from examples to generalizations and back again. Therefore, viewing a single object at a time in its entirety is the most natural browsing style. And in any event, this coincides with the familiar information retrieval model, in which users navigate from, say, web page to web page.
- **Category-based**. As the user browses from instance to instance, they identify similarities and relationships, grouping things into categories. The system reasons about the objects that have been so grouped, and begins to form statistical hypotheses about what attribute values could be used to characterize them in general. The user can then begin to operate at a higher-level of abstraction, asking questions about the categories themselves as bona fide entities. "What other data records are similar to this group I've identified? How large is this group, relatively speaking? What's the relationship between these two groups – do they overlap, or are they mutually exclusive? What characteristics are significant about the area of overlap? And what other data records out there are quite unlike anything that I've categorized thus far?"
- **System-directed**. In general, as the user explores the information space and begins to form judgments about it, the system will recommend which instance to look at next. This is based on a combination of two factors: (a) what the user is likely to find most relevant, and (b) what is most likely to reveal further information about how the user is trying to structure the data. User feedback is

[1] An "instance" here could be though of as a web page, an XML data record, or a database tuple, depending on the environment. The important thing is that it represents a single coherent object of study, probably composed of many attributes.

given to data mining algorithms that can make statistical predictions about these criteria. Users, however, can explicitly refocus the search at any time.

- **Individual**. Every user brings their own view to an information space, and hence the system permits them to structure it in a personalized way that is independent of anyone else. There is no one "right" semantics for any data: a user may categorize it in ways different than the original authors envisioned. Each task and role involves special needs and different interpretations, which the system facilitates and even encourages.[2]
- **Persistent**. The information gained from a user's interaction with the system is captured and stored. When the user returns, it's not up to them to "dust off the cobwebs" and try to mentally reconstruct what they had learned; rather, the system remembers the categories and relationships that were previously identified, and can readily produce them. This nascent structure can also be shared and compared with other users.
- **Augmentative**. As the user browses the data, they endow it with semantics. In some cases, these semantics may be broadly applicable beyond just the user who identified them, and it may be worthwhile to annotate them explicitly in the data. The system assists the user in determining when a perceived relationship is worth being denoted in the database proper, and provides an easy mechanism for doing so. In this way, a system like this could be the platform for the collaborative, evolving discovery and organization of a shared information space. As more users explore the system, more semantic annotations are added, and the better the system is able to identify real-world relationships.

Note that these goals are quite different than what is provided by today's popular interfaces. The system is not centered around concrete queries; these only come later once the user has a feel for the nature of the data and knows what questions to ask. The end goal here is not a task more efficiently performed, but an information space better understood. Intuition built, not time saved, is in view.

[2] This is a departure from the current "monolithic ontology" approach advocated by the majority of the Semantic Web community, as explained below.

Table 1 contrasts the data organization model with the query-response model, giving example settings of where each is appropriate. Note that today, the query-response model is being used to fulfill even the conditions listed in the *left* column, because it is the only available option. It is, however, rather ill-suited for them.

Best for the data organization model	*Best for the query-response model*
The user wants to really get a feel for the overall information space: an intuition for what kinds of records it contains, and how they are related to each other.	The user is interested only in isolated queries in one part of the space. They want to find a record or two, and then use them for some other purpose.
The main goal is a deeper and enduring understanding of what is represented in the database as a whole.	The main goal is the immediate answers to some explicit questions about one aspect of the database.
The semantics are in the eye of the beholder; the information space can be carved up in many different "correct" ways.	There is a single "correct" classification of the data, and lots of offline processing time can be used to learn it.
The user approaches the information space as a relative novice, unsure of exactly what kinds of things it contains.	The user knows exactly what the information space contains, and how to get what they want. The main point is just to get an answer quickly.
The user doesn't know exactly what they're looking for until they find it.	The user knows exactly what they're looking for *a priori*, and just needs a fast answer.
There is no one "right" answer to the user's implicit question; evaluation is subjective.	There is a right (and wrong) answer to the user's question; evaluation is objective.
Early in the query process, the user wants "some" representative records, and is content with finding some that look fairly satisfying.	The user demands that *all* the pertinent information, and *only* that information, is returned, regardless of where it is in the database.
Data records will be frequently added to the database; when they are, this may change the way the user conceptually categorizes the information therein.	Data records will be frequently added to the database, but any queries the user originally created will still stand "as is" in light of the new additions.
There are potentially many uses for the data besides what its authors originally intended.	The authors of the data know exactly how it is to be used, and may actually want to prevent it from being mistakenly used in other ways.

Table 1. Comparison of settings in which the two database interaction models are most appropriate.

1.4. The opportunity is now

Numerous researchers have recently been attempting to extend the World Wide Web into Tim-Berners Lee's vision of a "Semantic Web."[18] This involves the addition of unambiguous semantic annotations to traditional human-readable documents so that the web becomes machine-readable, and subject to automated reasoning. The idea is that the web, which is currently comprised of mostly natural language information, cannot be

exercised to its full extent until computers can safely determine its meaning without needing a human interpreter in the loop.

The primary intent of these efforts is to enable wide-scale *deductive* inference. The annotations that adorn web pages will make assertions about the real-world entities to which they refer, and these annotations will be selected from standard "ontologies," or declarative representations of particular domains. An ontology not only unambiguously defines terms and their allowable relationships, but also declares axioms that rule-based reasoning engines can use to infer additional truths from those explicitly declared on a page. The hope is that by incorporating this commonsense reasoning, machines will soon be able to automatically execute certain tasks that currently require a human to orchestrate.

It seems to me, however, that this annotative information can be put to another use entirely: namely, as the basis of *inductive* reasoning. If machines have access to structured, clear-cut assertions about entities and their relationships, then they have essentially the same kind of material to work with as relational databases, and the inductive data mining techniques described above can be applied to them. This is potentially much more dependable than current natural language processing techniques, which are forced to make guesses about the "meaning" of a page based on the frequency of words therein. As more and more domains produce Semantic Web-compatible information, they are unknowingly providing material for an interface like the one I'm describing to operate on. This may open up a whole new use for the Semantic Web than the original architects intended – not a "fire and forget" technology that proactively accomplishes tasks for you, but simply a richer and more dependable network of information that can be interactively mined. This rapid emergence of trustworthy data makes me believe that the time is ripe for the kind of interface described here.

The requisite machine learning technologies, too, are in place. Semantic Web annotations are inherently *graph-based*, expressing named relations between complex objects, rather than simply lists of atomic fields. Data mining has traditionally concentrated on tabular data with very little embedded structure, but recently work has been done to adapt these principles to operate on graphs instead[80, 98, 115]. The notion of "similarity" between concepts in a graph can be incorporated as a fundamental

building block of my interface. Also, the emerging sub-field of "active learning"[105, 127] is immediately applicable to my work. In this paradigm, rather than a system passively accepting a set of classified examples to study, the system plays a role in actively choosing which examples will best help it learn. This task is essential to the user interaction experience described above, since the system suggests what instances to show the user, and its aim is to learn a definition for the relevant category. By incorporating active learning principles, the system can select optimal instances for the user to examine, improving its learning power.

In conclusion, the system I propose satisfies a currently unmet need in human interaction with large information stores. And now is the perfect time to introduce it. The growing enthusiasm for the Semantic Web ensures that plenty of structured data will become available for such a system to operate on. The underlying techniques necessary to support its operation have been developed well enough to be applied in this new setting. It remains only for someone to pull together these three fields – human-computer interaction, data mining, and the Semantic Web – to result in a rich and powerful tool for increased human understanding.

2. Related work

2.1. Competing research

Other researchers have attacked various facets of the problem outlined above; however, none of them address it in its entirety, and there are important differences between their approaches and mine. In this section I identify the principal publications I have found in the relevant areas, and point out what makes my own work distinct.

2.1.1. The Semantic Web

As mentioned above, the major thrust behind mainstream Semantic Web research is to enable deductive reasoning. Much work has gone into languages for crafting ontologies, methods of annotation, inference engines, task-achieving agents, and various supporting technologies[33, 35, 43, 126]. None of this work bears any resemblance to

my investigations. This underscores the fact that the inductive approach I prescribe is certainly not prevalent within the community.

Various navigation and browsing tools have been developed to "surf" this Semantic Web. These systems often incorporate an ontology directly into the user interface, and aid the user by automatically generating hyperlinks to the related objects that are specified in RDF annotations[3]. The most basic of these is SHOE Search from the University of Maryland[58]. Users explicitly choose an ontology and then a class against which a query should be issued. The system presents a list of properties for that class; the user selects one or more values, and then gets a list of hits which may include URL bindings. This kind of system simply raises the search process to the level of precision made possible by the underlying semantic data.

McGuinness's FindUR[85] and more recently Davies's QuizRDF[36] elegantly combine an ontology-based semantic search with a free-text search, allowing users the freedom to migrate to Semantic Web format gradually. These systems integrate browsing and search functions: the user can type free text to be searched for, and also traverse relevant taxonomies to find instances of certain types directly. FindUR is aimed to help naïve users form more effective queries by showing them the ontology and hence the most appropriate search terms. It uses subsumption (superclass/subclass relationships) to increase recall. QuizRDF incorporates semantic assertions into a multi-dimensional indexing scheme, so that queries can include precise ontological terms.

Similarly, the Spectacle system[47] allows users to select and manipulate taxonomic terms in various ways to find instances. A visualization tool is included that shows where certain terms are matched to documents graphically. "Cluster maps" allow the user to select terms of interest and visually inspect which documents include various combinations of those terms. This allows users an objective glimpse into the information space structure.

The SEAL portal from the University of Karlsruhe[82, 116] sits on top of an inference engine and associated knowledge base. Its navigation module automatically generates links for all related instances of a particular page, based on the knowledge base assertions that mention that page. A query module puts an easy-to-use veneer onto the

[3] See the appendix for a brief summary of the relevant Semantic Web technologies.

engine's F-logic query interface. This facilitates the notion of "query links," or hypertext links containing dynamically-evaluated queries that are executed when clicked. Ontology-sensitive query forms provide semantic browsing capabilities similar to the systems mentioned above. More interestingly, SEAL performs a "semantic ranking" of query results based on the perceived semantic similarity to the posed query. This incorporates ideas such as the relative proximity of terms within a taxonomy.

All of these efforts introduce semantics into the search and browsing process in one way or another. However, they are nearly all simply straightforward ways for the user to view and traverse the static ontology that the data was defined against. None of them include a predictive or learning component; none permit a user to specify their own groupings of objects and make discoveries outside what the data authors intended. Spectacle addresses the notion of user individuality, but this is simply to trim down a full ontology (viewed as too complex for the average user to comprehend) into a lighter-weight taxonomy better suited for a particular type of user. SEAL features a semantic similarity engine, but it is used only to rank results, and uses a fixed notion of similarity that does not adapt to user perception. Spectacle's cluster maps are configurable, in that the user can choose the terms upon which they will be based. However, the system does not assist them in discovering this: it is up to the user to know in advance what features are relevant for the relationship of interest. Finally, there is no notion of first-class categories in any of these systems: the user cannot explicitly discover them, refine them, or ask questions about them.

In general, these systems can be viewed as early investigations into how the Semantic Web can best be navigated and understood. The subjective, system-directed, category-based method I propose is an alternative approach to this.

2.1.2. Task-based systems

The idea of user individuality has been explored, especially as regards task-based or role-based systems. Here, it is acknowledged that each user has different information needs and perceptions based on the particular task they need to accomplish. Systems therefore need to be aware of this context, and adjust accordingly. In his excellent survey, Shahabi[106] identifies three techniques for personalized search systems: (1)

Personalized page rankings, where the interests of users are taken into consideration in filtering query results (these interests are either explicitly stated, or inferred from favorite pages of users.) (2) Query refinement, where the same kind of information is used to insert additional terms or otherwise refine a query before it is sent to an engine. (3) Personalized metasearch systems, where results from multiple search engines are aggregated and the results weighted based on the user's perceptions of the features and the engines themselves. Two things are to be noted here: first, all these approaches are heavily steeped in the query-response paradigm, and second, the emphasis is on sifting "irrelevant" information from what the user finds "interesting." In my approach, categorization is not a simple matter of interesting vs. not interesting – it is the discovery of different *kinds* of interesting things, and how those different kinds relate to each other.

The EnerSearch Industrial Research Consortium bills itself as a "virtual research organization," and is an early adopter of Semantic Web technologies.[60] It disseminates numerous energy-related reports on a wide variety of topics from a central knowledge distribution point. Architects have recognized the dichotomy between how information suppliers and information seekers view data structurally. For instance, providers view the space in terms of projects and authors, while seekers tend to be interested in a particular industry domain. As such, EnerSearch has worked to identify different target user groups so that interfaces customized to each group can be provided.

Boeing has taken a similar approach with their aerospace customer support system.[62] Support personnel rely on dozens of different resources scattered throughout the company in order to address the concerns of airline operators. There are many different types of problems they must solve, and it is recognized that different tasks require different resources. Hence, in the query interface to the enterprise-wide database, typical problem categories are enumerated, and the user may select one or more to help focus the search. Each task category or combination has a set of associated resource types that are deemed relevant to that category, and this gives queries more precision.

Both EnerSearch and Boeing address the subjectivity question introduced earlier, but only partially: they assume that an information seeker can be categorized into one of a fixed set of types determined *a priori*, and that the worldview of this user can be adequately described by the group as a whole. My system is different: I don't attempt to

make any advance judgments about the ways in which users might be classified, but rather let the system adapt to each individual uniquely.

Staab et al.'s "Smart Task" project[120] has the ambitious aim to imitate a human assistant, observing what the user does, answering incomplete questions, and forming expectations about what questions are likely to be asked next. Upon further inspection, however, it takes a very different approach than what I propose. Staab models business processes explicitly in an ontology, and provides annotated document templates that can be linked to specific tasks in that ontology. A reasoning engine operates behind the template interface and proactively helps the user fill out the fields. For instance, if a project manager wants to send a fax to all engineers who know XML, in order to about their availability for an upcoming project, the system executes a semantic query behind the scenes, and provides dynamic cues to the user as to the size of the resulting hit list. If a large number of employees have the requisite experience, the system may perform some pruning to include only employees who show an availability during the upcoming project's timeframe, but if only a few satisfy the query, it may return the list in toto. The results of the query are then automatically inserted into the template for convenience. This system also implements some performance boosting, by pre-fetching (and pre-computing in the reasoning system) queries that the agent expects the user to pose. This is all very primitive, however, and hard coded to the business processes encoded in the ontology. And at any rate it is all based on the query-response model.

2.1.3. Adaptive browsing

Several researchers have explored the idea of adaptive browsers, which observe user browsing patterns and make helpful predictions about what to look at next. Lieberman's Letizia system[73] is billed as an "information reconnaissance agent." It performs a continuous breadth-first search of the hyperlinks emanating from the current page, examining these new pages and making recommendations about what the user might be interested in next. These recommendations are based on a machine learning algorithm to measure degree of similarity to what the user has looked at before. This has some features in common with my work, but it does not offer a method of categorization per se. Its goal is simply to streamline the delivery of "interesting" material. And note

the dependency on the Web link structure between pages – it is powerless to find or suggest similar pages elsewhere in the information space. It is therefore intrinsically hardwired to the data authors' intentions: if the author of a web page foresaw the relevance of another page and linked to it, Letizia can recommend it, but otherwise, it is blind. Lastly, since these and similar tools were pre-Semantic Web, their calculations are based on natural language rather than on structured data.

Closer to my scheme is the work of Grosky et al.[53], whose system observes a user's browsing path to infer semantics. Their system is tailored to multimedia environments in which they seek to identify semantic similarity between images. The assumption is that over short distances, a user's browsing path will demonstrate semantic coherence and will reveal similarity between pages. Therefore, the semantics of a page can be derived statistically by analyzing the browsing paths of users toward the page. Like me, Grosky et al. recognize that a user may have in mind concepts too abstract to articulate, but that these concepts may be reflected in low level features that a data mining algorithm can detect. They also agree that the author of a web page cannot completely define that document's semantics, and that semantics emerge through use. One principal difference between this work and my approach, though, is that it is all "behind-the-scenes." None of the semantic similarity notions are surfaced to the users. Users do not guide the process, interactively organize the information space, make corrections, or anything else: they simply browse while the system "spies" on them and makes deductions. The system can then provide suggestions about pages it thinks might be of interest, but since categories proper are not presented as an object of study, many of the functions I aim to implement are simply impossible.

2.1.4. Information space organization

Tools specifically designed to help a user organize an information space are not completely unknown. They can be roughly divided into two types: objective, and subjective. Objective systems look only at the actual properties of pages (or other types of instances) to determine how they are structured and related. Hence, an information space will be organized in exactly one way, no matter which user is exploring it. Subjective systems, on the other hand, take into account each individual's unique view of

the data, incorporating explicit or implicit feedback from the user to shape the organization.

Objective systems

Some advanced World Wide Web browsers perform natural language based clustering on pages that they find, in order to give the user a feel for what pages exist on a particular topic and how they are related. One example is the KartOO visual meta search engine.[12] It takes Google-style queries as input (simple sets of terms) and generates a cluster map of matching pages. KartOO automatically extracts other common terms from within the retrieved documents, and displays them in a two-dimensional graph so that the user can see at a glance which pages feature which subsets of terms. Grokker[1], a commercial solution from Groxis, offers similar functionality, though the visualization tool they provide is somewhat more sophisticated. It offers hierarchical subcategorization and some amount of user filtering to customize result sets. The aforementioned Spectacle interface[47] can also be considered in this category, except that it is a true Semantic Web application. It bases its clustering on shared ontological terms rather than natural language data. The work of Le Grande, et al.[69], is also akin to this, though theirs is based on Topic Maps[3], an older, alternative Semantic Web technology that seems to have fallen out of vogue. All of these solutions are objective, organizing the space automatically based on the raw data rather than on the relationships that users identify.

Liu, et al.[76], have proposed a technique to mine topic-specific knowledge on the Web. Their goal is to help people learn in-depth knowledge of a particular topic by compiling various web pages dynamically into what is essentially an electronic textbook. They use natural language processing techniques on search engine results to identify the key definitions pertaining to the topic, and the various sub-topics of which it is comprised. The result is an organized set of the most relevant pages that can be browsed to give an overview of how that topic is organized.

In a somewhat different vein are various attempts to automate the ontology engineering process by automatically discovering ontologies from text. Typical of these efforts are the Text-to-Onto system[79], which finds sets of frequently co-occurring

words and infers conceptual relations between them. Statistical techniques like these are incorporated into a larger, iterative process where users interact with the system to help settle on the best ontology for a particular domain. Hence these methods do incorporate user feedback. However, the objective is to determine "the" ontology that can serve as an authoritative structure for many systems to subscribe to, rather than honoring the unique organization model that every user brings to the data.

This has much in common with the emerging field of "web mining," or applying data mining methods to the content, structure, and monitored usage patterns of web pages. Cooley, et al.'s survey[31] provides a good taxonomy of this field. The Web->KB system and subsequent additions by Ghani, et al.[51], demonstrate how general patterns can be identified from heterogeneous, free-text web pages. They first use a variety of information extraction techniques to create a flat feature list, and then employ traditional data mining algorithms to draw general conclusions about the world that the Web represents. Berendt, et al.[16], give a good overview of how web mining methods are being extended to the Semantic Web. She encourages semi-automatic approaches, with user interaction to help guide the process, but again the accent is on learning a global ontology that characterizes the data, not a particular individual's view of it.

Subjective systems

The most basic approach to subjective information space organization is the now ubiquitous "bookmark," which allows users to index pages of interest and then organize them in a hierarchy of folders. The shortcomings of this idea are obvious – indeed, it is really just a more convenient way to perform manual categorization. The burden is squarely on the user to discern the correct categories and their relationships.[4] Among the first to suggest the bookmark technique were Oostendorp, et al[90]. They mention the idea that such a system could monitor frequency of usage and periodically suggest appropriate content reorganization, but this idea remains undeveloped.

[4] And hopefully the correct categorization will emerge very early in the process, else a rather common phenomenon is likely to ensue. Users often navigate quickly through the World Wide Web, frequently bookmarking pages of interest that they want the freedom to return to later. The result, however, is that at the end of a browsing session, the user is disheartened to discover that they have unwittingly created a large, impenetrable, flat bookmark space, which itself might be a good candidate for information retrieval! I suspect that most frequent Web users have had this kind of experience.

Chakrabarti, et al.[26], take this idea a step further, and attempt to mine "themes" from bookmarks. By collecting statistics on an entire community's bookmark folders, they mine for which words tend to occur in which documents, and which documents in which folders. This helps to expose themes: areas of topic interest within the community. Note that users are playing no active role in organizing the information space: the system is attempting to do this for an entire group of users in an offline process.

Wexelblat's "Footprints" system[133] is one of a number of systems that monitors user behavior and attempts to learn from it. Footprints observes the browsing paths of many different users who visit a site, and forms an aggregate understanding of what kind of navigation experience a new user is likely to undergo. It then makes available a graph for new users, allowing them to learn from what previous users experienced when they first enter the site. The aim is really to capture the user's cognitive model of how the space ought to be organized, rather than just the web site designer's model. This is a step towards true subjectivity: user behavior is incorporated directly into a learning algorithm, but it is still a "one size fits all" approach in that behavior from diverse users is aggregated in order to determine a sort of "average" browsing pattern for future users. Cooley et al.'s WEBMINER system[31] and Pirolli et al.'s "information foraging" theory[93-95] are also in this vein.

Although the overall goal is somewhat different, the OntoShare system of Davies, et al.[34], incorporates some degree of subjective information space organization. One of their interests is usage-based ontology evolution. OntoShare is an ontology-based World Wide Web knowledge sharing environment, which facilitates information sharing within "communities of practice" (i.e., users interested in a common domain, either inside or across organizations.) Each user has a profile that contains a set of ontology-based topics, specifying their set of interests. When a user posts a new document, the system suggests concepts that it might fit under (each concept is associated with a set of key words and phrases, and this definition is ever evolving.) In cases where none of the existing concepts is appropriate, the user can extend the ontology to accommodate the new document. In this way, the community evolves the ontology over time, collectively. As above, however, the entire community subscribes to a single view of the ontology even though they all contribute to defining it.

Boley et al.'s WebACE agent[20], however, realizes bona fide subjectivity. It observes user behavior (specifically the number of times a page is visited and the amount of time the user spends viewing it) to infer which documents are truly of interest. Then, it reduces each "interesting" document to a vector representation and performs clustering algorithms on it. These clusters can be used to generate queries for "similar" documents. Newly found documents can be filtered, re-clustered, or manually manipulated by the user. This bears some resemblance to my approach, except that (1) it is all based on natural language processing techniques, rather than structured semantic annotations, (2) there are no higher-level category operators that can be used to ask more abstract questions, (3) the notion is again of "interesting" vs. "not interesting" rather than grouping objects according to perceived similarity, and (4) the user is not helping the system make categorization decisions directly, but is rather relying on the underlying statistical processor to make the correct judgments for them. In fact, the user never even explicitly tells the agent that a document is "interesting" – this is inferred based on viewing behavior. This may give a user less hassle, but also less control.

Probably the closest competing approaches to my own work are the DASHER project from the University of Southern California[89, 139], and Simone Santini's "El Niño" system[103, 119] for emergent semantics. In the former, users collaborate to organize a web information space, creating hierarchical categories on the fly to which specific URLs can be added. Categories are like bookmark folders with a plus: they can be merged, all links emanating from a page can be automatically imported, etc. Natural language processing algorithms help identify common noun phrases in documents, so that users can compare their own categorizations with the noun phrase clusters. Category hierarchies can even be imported from sources like Yahoo, to serve as a starting point. The organization decisions that a user makes can be saved, named, and shared collaboratively.

The system assists the user in creating categories by citation indices that list the most frequently mentioned noun phrases. Subcategories are extracted by looking for words that appear to modify a noun that had been previously identified as a category. Inductive learning then attempts to populate categories with instances based on "bag of words" comparisons. The system builds decision lists to classify documents, for example,

"if a document contains word X and Y, then put it in category Z, else…" Users find these kinds of rules easy to understand, and they are readily transformable into Boolean queries. Finally, the system uses a "fan out" technique to examine the hyperlinks emanating from a page to search for other "similar" pages.

Despite DASHER's similarity to my proposal, there are important differences. The most obvious is that my work is aimed at exploiting structured, semantic annotations rather than less reliable free text. But even within the user interaction model there are distinctions. For one, DASHER does not permit the more complex category-level operations I envision. Although instances (here, web pages) can be grouped, and the system gives some help in doing this, one cannot ask questions specifically about categories such as estimated size, degree of overlap, broadening or narrowing options, correlations with other categories, etc[5]. Also, a category in DASHER is composed only of examples, not counterexamples, which makes it more difficult to specify a structure counter to the consensus view of that domain[6]. In general, the system is more focused on quickly finding information on a defined set of topics than on exploring the relationships between categories of instances throughout an information space.

Santini's work[103, 119] focuses specifically on image databases, and how the "meaning" of images is revealed as users explore, compare, and relate them. The El Niño system is a graphical editor that allows users to position images at various locations on a screen and thereby manually cluster them. Images from within the large database will automatically appear and disappear from the screen as the user interacts and the system learns which images are the most relevant. The idea is that the semantics of an image is not an intrinsic property that can be filtered via a query process, but rather an emergent

[5] In fact, since DASHER's domain of operations is simply "the Web" rather than a more focused information space, it is unlikely that such operations would make much sense. It is strategically relevant to know that 23% of one's workforce has a certain skill mix, or that 23% of one's collection in a video store is comprised of violent epics. But knowing that 23% of software-related pages on the Web refer to Java doesn't say much except about the trends and popularities among web authors.

[6] The induction algorithms are simply at the mercy of the noun phrases that tend to occur together. For instance, suppose a user wanted to define a category of pages relating to the New York Yankees, but which did not offer merchandise. One could identify many examples of pages in the envisioned category, but when judging unseen pages the system would find it difficult to avoid (and indeed, would not know to avoid) pages offering goods. This is simply because these concepts tend to occur together: pages about the Yankees do in fact tend to offer merchandise, so the category becomes muddied with unwanted sources. The only way to refine this category definition to be more suitable is to specifically *exclude* certain concepts. My method of doing this is for the user to specify *counter*examples to the category, which gives a machine learning algorithm the proper material with which to work.

property that user interaction exposes. Related images, once identified, can be manually lumped into a named "visual concept" for future retrieval, and decorated with metadata.

Some of the important differences between this work and mine include: (1) Santini's system is custom-tailored to the image domain, and to a visual process where clustering decisions are primarily reflected in pixel-by-pixel positioning on the screen. This makes it difficult to parameterize a category, to ask well-defined questions about it, or to share it outside the context of a particular user interface session. (2) It doesn't have a way to utilize defined semantics, or to participate in assigning them, instead relying on features extracted from image processing algorithms. (3) The system is not as multi-dimensional as what I propose, where instances can belong to multiple overlapping categories. In Santini's system, an image is either close to or far from another – it cannot be both, depending on the categorical context. This is due to the limits of the two-dimensional canvas upon which clustering is performed. (4) Again, the higher-level category-based operators so central to my system are not featured.

2.1.5. Information visualization

The field of information visualization can be distinguished from information space organization. In the former, the goal is to present data so that it can be dynamically navigated, while in the latter, it is to help the user form an enduring conceptual framework. Information visualization tools often make use of color graphics, animation, multi-dimensional displays, and other creative elements that allow users to examine and manipulate data visually and interactively. The idea is to let users freely explore a database along many different dimensions, and to recognize clusters, gaps, relative magnitudes, and outliers.

This is potentially a more powerful paradigm than the query-response model, since it incorporates such information-rich graphic elements. And it might easily be misconstrued as tackling the same problem as information space organization. But there are important differences. First, an information visualization tool typically provides a transient experience. It is rather like a powerful microscope with many different settings that permits a specimen to be viewed in numerous ways. The tool is powerful in what it can reveal, but at the end of the day the patterns that the user has perceived reside in their

mind alone – the tool did not attempt to explicitly capture them. In this sense, it is really just a fancy, interactive query-response model. Each time the user exercises one of the graphical functions, they are in effect adjusting and reformulating their query. It is still the user's job to assemble the various findings into a meaningful cognitive model.

With information visualization, too, the tool is passive, providing a variety of simple interactive elements that can be manipulated. The software makes no effort to "understand" what the user perceives. It does not proactively suggest what instances may be of interest next. There is no machine learning component to draw conclusions about groupings, nor is there normally any way to even specify categories: the data can be viewed solely along the primitive dimensions that define it, albeit in powerful ways. Information visualization can be seen as an attempt to provide the most effective means possible to browse data *in the context of the structure in which it was created*. It does *not* encourage the user to form their own alternative structure, identifying relationships and groupings that perhaps were not in the minds of the data producers, but which nevertheless may have great value.

Some of the most impressive work in this area comes from Ben Shneiderman and his colleagues at the University of Maryland[6, 110-112, 136]. These researchers have designed suites of tools that allow large data sets to be explored and compared along any of their axes. Visual displays may be expanded or shrunk, certain regions zoomed in on, data points color coded and/or filtered by any value or set of values. Time series data can be explored and filtered. Hierarchical displays can be created on demand, organized in countless different ways based on the features that users identify. These tools are all the more effective because the underlying rendering engine is so computationally powerful: this display updates smoothly in real-time, giving the user an incredible amount of flexibility in direct manipulation. Shneiderman's emphasis is on rapid, reversible actions[6, 111, 136] whereby a user can exercise great control over what is currently in view on the screen.

These efforts probably represent the pinnacle of what can be achieved in information visualization, while at the same time pointing out its deficiencies. A user can indeed explore data from any possible angle, provided they start out already knowing which angle is the most appropriate. They are equipped with many powerful instruments,

but the system offers no suggestions as to when and how to use each one. The onus is on the user to figure out which data to explore, which values to filter on, which features are relevant, etc. This is fine in situations when one has a good intuition about the structure of the data, and wants to see how the various instances map on to that structure. It is also satisfactory when one is content that the predefined structure sufficiently captures all of the meaningful relationships in the data, and that no alternative patterns of organization are of any likely use. But when either of these two conditions is violated, then these tools can only be used as the query-response model often is: to indirectly help accomplish something for which they were never designed.

Furnas and Rauch address some of these concerns in their NaviQue browser[49], which integrates components of both information visualization and information space organization. The tool's entire work surface is "live": anything can be typed anywhere, and any object used as part of a query. Every object on the work surface registers a vector of its contents with a "similarity engine" so that it can be reasoned about. The most relevant functionality to my work deals with the notion of "sets." For query processing, the user identifies a "query set" specifying desired samples, and a "collection set" which is the target of the query. The query set vectors are combined into a single vector, which is matched against all the information vectors in the collection set to yield a "return set" of best matches. The various kinds of sets can be freely composed, resulting in a sort of visual algebra. In this way, a user can express information structure through the objects they choose to include in various sets.

This system actually goes a long way towards satisfying some of the needs expressed in the introduction of this proposal. One of the chief differences between it and the system I propose, however, is that NaviQue was not explicitly designed for information space organization. The "sets" are very much like categories in their behavior, but the aim of the interface is not so much to create sets for their own sake, but in order to facilitate better querying. Hence their use as first-class organizational constructs are limited. They cannot be easily used to learn a category definition, since the underlying pattern matching used for set queries is not surfaced to the user interface. The user, therefore, has no visibility into the key components that give a category its conceptual coherence – a set is forever merely "these exact objects I've grouped

together" with no insight into the foundational relationship. One also cannot ask most of the higher-level questions about categories such as size, inter-category associations, other correlated attributes, and so forth. In general, since the aim of the interface is not to help a user systematize an information space, it inevitably falls short in supporting some aspects of this task.

Other factors distinguish NaviQue from my work, including the obvious facts that it is not based on Semantic Web annotations, and that it is so graphical in nature and therefore has some of the same biases as Santini's El Niño[103]. It also does not give users the ability and guidance to self-annotate data based on the relationships they perceive, so that other users can benefit from this. Sets are not designed to be shared, nor to automatically "fire" when new data enters the database. The system does not actively suggest beneficial navigation options, instead serving in a passive capacity as is typical of information visualization systems. Finally, NaviQue does not make recommendations about category membership. Granted, there is an "auto file" feature, whereby a new object can be compared to items already in a user's workspace and the most similar items flagged. But this is limited in two ways: first, each object is inevitably "filed" in only one place, which makes it tree-based rather than graph-based[7], and second, the filing suggestions are based on similarity to individual objects, not to categories, which is of course limited.

2.1.6. Interactive machine learning

As noted previously, traditional data mining is an offline, time-consuming task that involves complex algorithms analyzing large volumes of tabular data. The expense of training time is usually justified by the optimization of classification accuracy. There have been attempts, however, to involve the user in the process in various ways, and some of these are worth mentioning here.

One reason to involve humans in the machine learning process is to simply improve and speed up the operation. The idea is that users can lend their intuitive knowledge to the problem and prevent some of the pointless inefficiencies that result

[7] I.e., an object is categorized in only one place, as if in a strict taxonomy with a single parent, rather than being a member of potentially many different groups.

from a blind, automated approach. Fails and Olsen recently applied this principle to a machine vision setting[40]. By relying on humans to perform some manual classification to guide the process, their "Crayons" system can create an effective classifier in minutes rather than weeks. Users notice where overfitting is occurring, and specify exactly those examples that will correct it. The interactive machine learning paradigm also permits the inclusion of a far greater number of features, since feature selection is performed "on the fly" as users interact with the system.

Another motive for incorporating user feedback is the theory that human intuition can help create better classifiers. Ware, et al.[131], and Ankerst, et al.[8], provide different ways for a user to experiment with a decision tree building process, lending their domain knowledge to the procedure to improve its effectiveness. An algorithm, of course, knows only the exact data points that it has been given, but humans have the ability to reason about the semantics of the features and their probable relationships, and can thus help guide the construction of the decision tree towards rationality. Ware, et al., observed that users do the best job if only a few lower-order attributes are sufficient to support good predictions. An additional benefit of this approach is that by advising the system as it constructs the decision tree the human will gain a much deeper understanding of the relationships in the data set.

Yet another, more mundane reason for interaction is simply that data mining is a technical topic that involves very specialized knowledge, and many users need assistance in applying the techniques effectively. The OIDM system developed at the University of Vermont[28] is a sort of online consultant which guides the inexperienced user through choosing an appropriate algorithm, getting their data in the right format, putting results into layperson's terms, etc.

What all these efforts have in common is that "data mining" is explicitly in the user's view. The stated goal is to "learn a classifier" or "shape a decision tree" or "choose and apply a data mining algorithm." In the system I propose, of course, all such notions are in the background: algorithms are being executed to assist the user, but the user is blissfully unaware that they are happening. Some of the principles, however, can be applied to my work, especially the Crayons system's example selection process[40]. The process of sharpening a category's boundaries, after all, requires the user to provide

information about certain critical examples, and this is a similar problem. Their interactive feature selection methodology is applicable as well, since identifying which attributes and values truly characterize a category is an important part of information space organization.

2.1.7. Recommender/matchmaking systems

Finally, it is worth noting that numerous systems have incorporated various kinds of similarity metrics to achieve tasks other than information space organization, usually retrieval or recommendation. For instance, Shardanand, et al., did this in the early days of the Web with their Ringo musical recommendation system[107]. Based on ratings of various musical artists, the system correlated each user to others who shared their tastes, and then recommended songs that their peers rated highly. This could be viewed as an impromptu clustering mechanism, based on fuzzy similarity. The GroupLens system[99] used the same approach for a different goal: filtering out voluminous content rather than recommending new content. GroupLens had users rate each newsgroup article that they read for interest and relevance. Then each user's profile of ratings was correlated with other users who rated items similarly, in order to predict new ratings for unread articles.

Such methods are ubiquitous in e-business, of course. Shahabi's overview[106] distinguishes between content-based filtering (recommend items "similar" to other items this user has bought) and collaborative filtering (recommend items that users "similar" to this user have bought.) Both are used heavily in e-commerce applications; Ringo was based on the collaborative model[107], while Amazon.com uses a form of content-based filtering[75]. Either way, the premise is the same: objects (either customers, or items) are effectively categorized according to strategic commonalities. Popescul and Ungar combine these two approaches[96], assuming that users are interested in a set of latent topics which in turn "generate" items and item content. They use an EM algorithm to learn model parameters, and increase data density by guessing which items the user likely accessed without the system's knowledge.

Ostertag et al.[91], demonstrate the validity of such reasoning in a completely different domain (software engineering), and show that incorporating domain-specific knowledge is sometimes necessary to achieve a high degree of accuracy. Their AIRS

system helps software developers find candidate components for reuse. Similarity heuristics based on the stated features of components facilitate a crude semantic-based search. "Similarity" is defined here as the expected effort to implement a desired component by using a candidate component. Their "subsumption metric" and "closeness metric" are specific to the software component reuse task, and attempt to take advantage of special domain knowledge. There are no ontologies used per se, but a thesaurus of synonyms helps resolve ambiguity and increase recall.

In any case, it is clear that some of the underlying machinery used by these systems is similar to mine. We both need heuristics for judging the relative semantic similarity between items with structured features. The differences are that in my system, the similarity metrics will adapt automatically in response to user feedback, and they will be used for an entirely different purpose: not merely recommending new isolated bits of interesting data, but forming a global understanding about an entire information space.

2.2. Ideas to incorporate

The scope of my research is broad, touching on many different areas of artificial intelligence, distributed databases, and human-computer interaction. I believe, however, that I can draw on the work of many other researchers in these various fields to make my job easier. In this section I identify some of the relevant efforts whose algorithms, approaches, or even implementations I can include as components of my contribution.

2.2.1. Machine learning

The popular machine learning algorithms that are available "out of the box" are too numerous to be cataloged here[56, 101, 138]. Suffice to say that many standard techniques exist for using labeled examples to create classifiers that will predict the labels for unseen examples. This is very similar to the basic problem of determining whether a new instance is a likely member of a particular category – one could consider the categorization problem as a simple classification task, where "in the category" and "not in the category" are the two labels. I will experiment with several classification algorithms – most likely including Naïve Bayes classifiers, decision trees, decision lists,

and support vector machines (SVMs) – to see which algorithm performs best on the categorization task.

One aspect of my system that is somewhat unique is the extreme scarcity of the data. As the user creates a new category and begins combing the space for examples, the system will have very small sample sizes with which to work. Hence, statistical confidence will suffer at first, growing as the user develops the category further. This may impact my choice of algorithms, as some may prove to yield better results with a small number of data points than others.

2.2.2. Active (machine) learning

Traditionally, machine learning calls for the compilation of as large a training set as possible, and then trains on *all* of this data, so as to build the most informed classifier. The idea is that the more the system knows at the outset, the better equipped it is to make judgments about newly arriving data. Needless to say, this is not the situation that my system will find itself in. The interactive, exploratory nature of the proposed interface means that time-conscious users will be rendering on-the-spot categorization decisions, limiting the amount of data available. It is crucial, then, to gather information about the most informative examples, so as to maximize the value of user feedback.

The field of active learning addresses just this situation. The premise is that instead of demanding a large number of labeled examples up front, the system has access to a pool of unlabeled examples, and can request the labels for some subset of them. Tong and Koller[127] present a series of algorithms that choose which examples will best help build an effective SVM classifier, and demonstrate that in some cases, the number of labeled examples required may shrink by an order of magnitude. Schohn and Cohn[105] go even further, and report that training on only a well-chosen subset of the available data, rather than all the data, actually yields a *better* classifier in many cases. This is evidently because only the most pertinent examples (close to the decision boundary) are taken into account, rather than large bodies of "obvious" instances whose noise may distract the SVM from what is truly relevant. These results are quite encouraging, and suggest that it is possible to focus the user's browsing experience on the

most productive part of the information space in order to deduce category definitions. I plan to experiment with these techniques in my implementation.

2.2.3. Similarity metrics

Many researchers have pondered how to measure the similarity between objects, since this is a necessary component for all kinds of fuzzy applications. The earliest pioneer was Amos Tversky, who attacked the problem from a psychological perspective in the mid 1970's[128]. He challenged the idea that perceived similarity is either symmetric, transitive, or the exact opposite of "different-ness," providing convincing counterarguments to all three properties. He proposed a non-metric scheme based on feature sets that has since been adapted in numerous guises to form the basis of nearly all similarity measures, including those of the recommender/matchmaker systems described above. Tversky advised that similarity should be based on both the commonality and the distinctiveness of features, with each of these components being weighed arbitrarily. He also considered the notion of context, and what leads humans to weigh some attributes more heavily than others. Though this work is old, nearly all the principles are directly applicable to my work, and I may adapt some of the ideas in a new form.

As a precursor to their image processing work, Santini and Jain extended Tversky's feature contrast model by applying fuzzy logic to it, and taking into account the perceived effects of interdependence of features.[104] These principles were applied specifically to the similarity of images, but they may have wider applicability.

The RDF data that my system will operate on is inherently graph-based, and various attempts have been made to measure the similarity between graphs, or between nodes in a graph. Some of the lessons learned can benefit my work. Rada, et al.[98], propose a metric for measuring the conceptual distance between semantic networks. They do not focus on a node's properties, but measure the similarity between networks as the average minimum path length over the pairwise combinations of nodes that they share. By introducing a factor that takes into account the similarity of nodes (rather than treating them as either identical or not identical), I may be able to incorporate this algorithm. Interestingly, Rada, et al., also take issue with Tversky's assertions that similarity is not

symmetric or transitive. They assert that by properly framing the problem, these properties can indeed be preserved.

Maedche and his colleagues have made similar progress, but in considering ontologies specifically, rather than semantic networks[80]. They consider relative locations within a taxonomic hierarchy to make judgments about how closely two terms are related. This is used in their "semantic ranking" of ontology-based query results. They have also developed a natural language processing system[79] that learns conceptual (non-taxonomical) relations between ontological concepts. This is based strictly on frequency of co-occurrence. Their method uses Srikant's and Agrawal's algorithm[115] for discovering generalized association rules at the optimal level of abstraction within the ontological hierarchy. Although my work is not based on free text, I may be able to take advantage of some of these findings in order to discover features that are conceptually similar. Objects that have been heavily annotated may have redundancies, and additional associations between objects may be inferred by co-occurrence frequency mining.

Strehl, et al.[122], have studied a variety of different clustering techniques and similarity measures for high-dimensional feature spaces. They systematically compared the performance of these algorithms in automatically classifying web pages. Empirically, the best similarity measures for this problem are the usual vector cosine measure ubiquitous in information retrieval, and the extended Jaccard measure, which judges the similarity between two documents to be the number of features they have in common divided by the total number of features they contain. The most effective clustering technique was weighted graph partitioning, in which a fully connected graph (the nodes being connected by weighted edges according to some similarity measure) is partitioned into disconnected subgraphs by the systematic removal of edges. The objective is to remove a set of edges whose sum of weights is minimized. This technique easily outperformed all others. Whether these conclusions hold in the lower-dimensional, more structured space of semantic annotations remains to be seen. But Strehl's findings are a baseline from which I can work, and the issues he discusses are mostly pertinent to my implementation choices.

Probabilistic relational models (PRMs) are an extension of standard Bayesian networks that can represent graph-based phenomena[50]. They describe a probability model for classes of objects, and for the relationships between objects, rather than merely for atomic attributes. Methods exist for learning PRMs directly from relational databases, and it should be easy to adapt these techniques to RDF data, provided some limitation is put on the lengths of the paths considered. A PRM can then be used to predict whether two objects will be related to each other, and which attributes and dependencies play the biggest role in that determination. Potentially, the PRM construct could be foundational to my implementation.

Finally, methods from the field of case-based reasoning (CBR) may be applicable[132]. At the heart of every CBR system is a case retrieval function that uses some kind of similarity metric to search for past cases related to a new one. Much effort has gone into optimizing these comparisons. The techniques are generally not new – nearest neighbor retrieval or ID3 decision tree induction, quantization and scaling of non-numeric attributes, feature weighting – but the literature includes a wealth of pragmatic lessons that have been learned from various industry domains. CBR systems have been developed for domains as diverse as customer service, medical diagnosis, electrical circuit design, gourmet cuisine preparation, and the resolution of legal disputes. Each of these arenas has its own data peculiarities and rules of thumb. After I have settled on a domain for which to specialize my system, I plan to scour the available CBR literature in that area for insights into how the data is typically modeled, and how best to judge similarity between the relevant domain objects.

In sum, there are many open questions as to how a system like mine should judge semantic similarity between objects. This will be a matter of experimentation, to see what underlying algorithms lead to the most satisfactory user experience. This subsection merely demonstrates that most of the general issues have been investigated before, and that I have a wide array of established techniques at my disposal to consider and possibly include.

2.2.4. Retrieval by Reformulation

Some of the human-computer interactions I envision bear a resemblance to the "retrieval by reformulation" work of Michael Williams[135]. His RABBIT system was designed to explicitly model the cognitive process that nearly all information seekers go through. When a user engages in an information retrieval session, they inevitably iterate and repeatedly refine their query in response to the information they receive. The successive examples they encounter are the crucial cues to help them discover what they are really looking for. RABBIT is a portal to a structured database, and gives the user the chance to openly critique the returned information. Users can select individual fields of displayed items and either "require" or "prohibit" them. This tells the system that a particular field value is either exactly what they're looking for, or exactly what they're *not* looking for. The system then finds another example based on this new information and shows it to the user. The idea is that the system will rapidly converge on the desired data, since every wrong turn is corrected and every hint of progress encouraged.

This process of critiquing and successive refinement is similar to mine, though RABBIT does not proactively reason nor is it designed for information space organization. But the principle of allowing users to comment on individual fields is a good one, and can only lead to increased comprehension of the information space. Williams also has an interesting idea of customizing a user's perspective of a complex object (ie., by restricting the fields shown) based on their query descriptions. This especially facilitates novice and casual users who don't yet know what the database contains. I believe this could be expanded such that the system chooses to reveal additional features of objects when it determines that these are relevant for category definitions. This allows a new user to "ease into" what might otherwise be an overwhelming amount of data.

Finally, Williams' notion of "analysis functions" may be beneficial. These are essentially custom filters that can be applied to various collections of objects, yielding tables, plots, graphs, etc. A similar idea could be incorporated into my category-level operators, to add visual feedback of trends and correlations.

2.2.5. Information gain evaluation metrics

Evaluating the performance of my system may prove to be as difficult as building it! How does one go about measuring "the depth of understanding of an information space" in a meaningful way? One idea is to track the rate at which instances are assimilated into user-defined categories; that gives an idea of the percentage of the overall space that the user has identified some meaningful abstract properties of, and how fast that percentage increases.

Another way is to incorporate some ideas from Pirolli's and Card's information foraging theory[93-95]. They model an information space as a number of discrete "patches" of information, each of which contains some dense amount of interesting material, but which will be depleted as the user continues to explore it. The theory states that users subconsciously weigh the tradeoff between staying in one patch too long and venturing to find another patch, which takes time and energy. The best any tool can do is maximize the user's rate of information gain. This paradigm may apply quite nicely to my system. A user will typically be focusing on one category at a time, and the search for items that conceptually match this category could be viewed as scavenging within a single patch. Changing categories involves a mental shift and a new search for a different mental construct, and could be viewed as moving to another patch. By comparing my system with a more conventional database interface (ie., Boolean queries on combinations of attributes) I may be able to gather some quantitative metrics to evaluate how much faster (or slower) a user assimilates meaningful information with my approach.

2.2.6. Semantic Web tools

Lastly, note that the Semantic Web is mature enough to have given rise to a considerable amount of freely-available support software. I will not have to build everything from scratch, from the annotations on up. For instance, numerous RDF triple stores are available, including "Redland," a robust library and multi-language API that allows RDF graphs to be parsed from XML, stored, queried, and manipulated[15]. In addition, the RQL language first proposed at the ICS-FORTH Institute in Greece[63] is a simple but effective declarative query language for Semantic Web annotations. It is

gaining popularity within the industry and may emerge as a standard. To satisfy higher performance needs, researchers at Vrije Universiteit in Amsterdam have made their Sesame architecture[24] publicly available. This is a flexible infrastructure that allows a wide variety of underlying storage mechanisms and interface protocols to be "mixed and matched" so that RDF statements can be accessed. In general, the World Wide Web Consortium[2] and other Semantic Web advocates are an open and collaborative group, and it appears likely that I will be able to find ready implementations for some of the more mechanical parts of my system.

3. General approach

In this section, I briefly outline my initial approach to the problem of information space organization. Again, I emphasize that all of the following ideas are preliminary. In fact, one of the primary goals of this project is to *discover* the best user interaction methods for such a task, so it would be presumptuous of me to assume that I had identified the perfect solution at the outset. Nevertheless, I have some ideas that I will start with, and this section lays out the best of my current thinking.

Instead of concrete diagrams, I will describe the interface's principles and workflow in more general terms. This is because the way in which the interface concepts will actually present themselves in on-screen widgets is still far from being determined. Suffice to say that it will almost certainly be graphical, where the user can directly manipulate the basic constructs via mouse gestures.

As previously mentioned, the user interaction model will be based on the concepts of instances and categories. I imagine two major "modes" being available: instance mode, and category mode. Users will seamlessly and naturally transition between these modes frequently during a session. The basic flow involves three processes: working from examples to a category definition, working from a definition to get correlations, and working from a definition to find new examples.

3.1. Instance mode

The user can browse from instance to instance, always viewing a single object in its entirety. The display will be simple and unobtrusive, merely presenting the object's set of properties and relations to other objects. At first, this will be an exhaustive display, possibly requiring scrolling if the object is sufficiently complex. Eventually, however, I would like to experiment with Williams' idea of presenting a custom instance view where certain fields are omitted from the display in order to hide unnecessary complexity. The idea is that based on the correlations that emerge, the system would be able to determine which attributes are truly significant, and which are clutter, in the context of an individual user's interaction. This idea is admittedly not yet very developed.

Although the instance display is simple, the choice of *which* instance to display is not. It depends on a combination of factors: (1) explicit user direction (including concrete queries to find a known instance), (2) the desire to show the user valuable information, and (3) the desire to flesh out a category definition by eliminating uncertainty. The interplay between factors 2 and 3 will be especially interesting. Active learning techniques will play a part, so that the examples given to the user to comment on are as productive as possible. At the same time, however, I am conscious of the fact that a human is involved, who is hoping to see mostly relevant information. Hence the goal will probably be to explore the boundary of the category *from the inside,* displaying probable counterexamples more rarely than probable examples. Much empirical evidence will be gathered as part of this study in order to determine what instance selection policy creates the most effective user experience.

Behind the scenes, there will always be an "active" category. This will either be a named category that the user has previously identified and is explicitly working with, or an unnamed, default category that will capture the user's current browsing context. When the user first engages with the system, a default category will be active. It will collect information and form hypotheses, ready to be named and made concrete as soon as the user so chooses.

As users browse instances, they identify them as either examples or counterexamples of the active category. My current preference is for this identification to

be explicit – a tangible interface event will mark the instance as "yes, this is the kind of thing I'm looking for" or "no, it isn't," after which another instance is shown. Another option would be to make such identification implicit, based on observed user behavior. The user can exercise more fine-grained control over the instance browsing process by explicitly requesting the system to require or ignore certain fields. This also helps the underlying machine learning algorithms focus on more relevant attributes.

Eventually, the user will have a number of named categories, and can freely switch between them as they browse. Any instance can be a member of multiple categories.

3.2. Category mode

The active category is always gathering information, even when the user is in instance mode, by assimilating new examples and counterexamples and analyzing their properties. At any time, however, the user can deliberately switch to category mode, and view the category explicitly. The system will also bring the category to the user's attention at other times, when it has something important to say about the conclusions it has reached.

A category consists of (a) a set of examples, (b) a set of counterexamples, (c) a "hypothesis" about what the underlying category definition is, and (d) a "confidence level" in that hypothesis. If and when the confidence level reaches a certain threshold, the system will surface it to the user and ask for confirmation. The user can browse the hypothesis, examining which fields and values the system thinks are significant, and possibly edit it based on their intuitive knowledge of the domain. At that point, the category becomes concrete and the user is encouraged to name it. If a category gains a certain number of examples without sufficient confidence gain, the system will tell this to the user and suggest further annotation to reflect the user's conception.

Once a category is concrete (or possibly, even before) it can be used as an operand to various category-level operators. This is the heart of category mode, in which the user can raise the level of abstraction from instances to entire groups and ask questions about their properties. Some operators that seem useful include:

- Find other likely instances. This operation actually resembles the query-response model, in that it gives concrete, instance-level results. The difference is that the query is based on a category, which was forged from a set of concrete instances already identified, rather than on an ad hoc set of keywords. This can be used to find additional objects of interest.
- Size. Once a category definition has been crystallized, a user can ask, "what percentage of the total instances in the database fall under this category?"
- Broaden/narrow. Given a category definition, the system can explore which of its criteria are the most and the least restrictive. This gives further insight into a category's size: a category may be small, but why? Which of its conditions are the most confining, and prevent the greatest number of potential members from meeting its definition?
- Set-theory operators. A user can create a new category from existing ones by employing standard set operations such as union, intersection, subtraction, etc.
- Find correlations. Once a group of instances has been identified – say, as the joint intersection of several different categories – the user can ask to see what additional features may be statistically significant about that group. This helps shed light on trends.
- Category comparisons. Any two categories can be compared to find distinctions between them. This may yield further insights about trends in the database. For instance, a feature may not be significant for either of two categories when each is compared with the whole population, but when the categories are compared with *each other*, side by side, a distinction may emerge between the two.
- Find unclassified instances. If the user's goal is to plumb the depths of the information space, they will want to move on to untapped soil once they're satisfied with the previous category. This operator focuses the user's attention on instances that they have not yet seen, and which do not fall under any of the categories they have yet created.

Again, these are preliminary, and more will surely be discovered as I work with representatives from a particular domain.

Finally, when new data is entered into the information space, existing categories "fire" if their definition is a match (or close to a match) to the new data record. Users are asked to confirm or deny that the new addition is a category member. This further strengthens the category definition, and keeps users appraised of newly-arriving data they may find interesting.

4. Detailed Scenario

To further crystallize these notions, I provide the following detailed scenario for the domain of workforce development. This gives some concrete ideas of how an information organization tool might be used in practice to accomplish real, practical tasks. It does not involve every feature of the interface presented in section 3. In fact, this scenario concentrates almost exclusively on "category mode" rather than "instance mode" functions. Yet I believe it gives a good flavor of the advantages of the paradigm.

I presented this scenario in its entirety to four representatives from Human Resources departments within the Colorado State government, and received enthusiastic feedback. The two main cautionary notes I received were (a) some of the data I describe may not be available electronically (yet), and (b) in parts of the public sector, employees may not be as readily transferable from project to project as I suggest. None questioned the utility of the overall scheme presented, however.

4.1. Getting a handle on a workforce

Stacy has just been promoted to a newly created position in a mid-sized software corporation[8]: Director of Strategic Staffing. Her mission is to take a broad view of the entire workforce, assess its core competencies and areas of weakness, and proactively decide how to shape it in order to enhance the future of the business. It is essential that she develop a firm grasp of the skill sets possessed by the workforce, and how they are distributed among workers in various roles. She incorporates advice from the Marketing,

[8] I choose to use an IT firm as my running example simply because that is the domain I'm most familiar with. Nothing fundamental depends on it; presumably nearly every organization has a mix of workers with quantifiable skill sets.

Advanced Research and Development, and Strategic Business Planning groups to anticipate what kinds of new projects are on the horizon, and what kinds of workers will be necessary to support them. It is her job to ensure that the business plans for current and future products do not fail because the company's resources are a poor match to support them.

Fortunately, Stacy has a comprehensive online database system to assist her in this task. In it, coded information is available for every staff member in the business, including:

- demographics information (employee's name, age, etc)
- company information (employee's department, seniority, etc)
- project information (which projects the employee has participated in, for how long, and in what capacity)
- skill set information (scores on technical exams, levels of competencies and years of experience with various technologies)
- performance appraisals (annual scores, targeted areas for improvement, etc)
- employee feedback (job satisfaction scores, career development goals)
- training (academic education, in-house training courses taken and planned)

Taken by itself, however, this data is nothing more than an overwhelming set of isolated facts. It does not automatically help Stacy accomplish her mission. What she needs are tools to help her navigate, understand, organize, and draw conclusions about this data so that she can perceive the trends present in the company's workforce and the challenges they pose. We will now see how she can do just that.

To help her get a feel for what kinds of workers the staff is comprised of, Stacy visits personally with several first-level managers. She asks them to describe their teams, what strengths and weaknesses they perceive, how they would evaluate performance on past projects, and so forth. She encourages the managers to be specific, to identify particular individuals and what each one brings to the table. Stacy makes notes about a few specific workers and emerges from these meetings with some understanding of how these managers perceive their staff's skill makeup. In particular, she identified a common theme from those she interviewed: a certain technology, called *web services*,

was perceived to be crucial to the success of a number of future projects. Stacy asked each manager for examples of workers who had demonstrated competency in this area, and jotted down four names: Walter, Wendy, Wanda, and William.

Stacy heads back to her office and logs on to the database. The system is designed to help her organize the company's workforce in various ways by establishing flexible categories that employees might fit into. Accordingly, she instructs the system:

```
Create a new category called "web services engineer."
```
[9]

The system establishes an empty bin, as it were. Then Stacy says:

```
Add Walter, Wendy, Wanda, and William to this category.
```

The records for each of these employees can be retrieved via a straightforward lookup by name or employee ID. The category now consists of these four employees. Now, Stacy executes the command:

```
Learn category definition.
```

The system analyzes the records of these four employees, looking for attributes that they have in common, and especially common attributes that are relatively distinctive compared to other employee records that have not been so identified. After a few moments, the system responds with:

```
Employees in category "web services engineer" tend to have:
   Skill : Object-oriented programming - moderate to advanced  AND
   Department - Engineering  AND
   { Experience : Java - 3 or more years  OR
     Experience : C++ - 4 or more years  OR
     Experience : Smalltalk - 4 or more years } AND
   Experience : XML - 2 or more years  AND
   Job Satisfaction : previous year - medium to high  AND
   Performance Appraisal : previous year - medium to high  AND
   Seniority - 3 or less years  AND
   Training : academic - B.S. degree  AND
   Training : in-house - none
Edit?
```

[9] The details of exactly how the user interacts with the system – whether the interface is command-line, point-and-click, or graphical; what kinds of widgets the user can manipulate, etc. – is left unspecified here since it is not germane to the present discussion. Instead, I include in the narrative only general descriptions of "messages" that represent communications between user and system. I trust that these messages are straightforward enough that the reader could imagine them being implemented in any number of different ways. Part of the implementation phase of this project will be to determine the best way to facilitate this communication.

The system is presenting Stacy with its unbiased findings about what seems statistically distinctive about the employees she has grouped together. She is now free to examine and edit this preliminary category "definition," which she does. Some fields that she recognizes as spurious she trims from the definition (perhaps by clicking on them and choosing an "exclude" menu option.) This instructs the system to ignore those fields for the purposes of identifying instances of this category. Examples might be the JobSatisfaction and PerformanceAppraisal fields. Although common across the four initial examples, these attributes do not really pertain to the notion she is trying to capture; namely, employees who possess a certain critical skill mix. (Their correlation is probably attributable to the fact that the managers tended to provide examples of their most highly-performing employees when asked.)

Some of the other findings are more eye-opening. While C++ and Java were frequently mentioned during her interviews with the manager, Smalltalk was not, and Stacy wonders whether Smalltalk is another applicable skill. She decides to leave this attribute in the definition for now. Even more noteworthy is the fact that the four employees identified have evidently been in the company only a short while, and that they have received no on-site training classes. Stacy removes these from the definition, but only after making a note to remind her to see if there are any related company-wide correlations later on.

After this refinement, Stacy now has a starting point for how to characterize an employee with a mix of skills that is evidently highly sought-after. She decides to see how this relates to the company as a whole. She commands:

```
Compute category size.
```

The system performs calculations, and soon responds:

```
There are 31 employees who can be classified in category "web services
   engineer" out of 4,981 total employees (0.6%).
```

This is striking as well! Evidently the skill mix she has identified is in short supply indeed. To get a more accurate percentage, Stacy decides to restrict the search to engineers, since that's really what's in view here:

```
Compute category size, relative to employees in Engineering Department.
```

The system returns:

```
There are 31 employees who can be classified in category "web services
   engineer" out of 1,255 total employees in Engineering Department (2.4%).
```

The original finding is confirmed: the vast majority of engineers do not have this kind of expertise. Which aspect of it is the most scarce? Stacy finds out:

```
Broaden category.
```

This tells the system to explore ways in which the category could be widened (i.e., so more employees would qualify as members) by relaxing its various constraints in turn. The system performs a good deal of statistical experimentation, and then reports:

```
Broadening category "web services engineer"...
   Allowing "Skill : Object-oriented programming - beginner" will include 71
      employees (5.6%)
   Allowing "Skill : Object-oriented programming - none" will include 636
      employees (50.7%)
   Allowing "Experience : Java - 1 to 2 years" will include 45 employees
      (3.6%)
   Allowing "Experience : C++ - 1 to 3 years" will include 36 employees
      (2.9%)
   Allowing "Experience : Smalltalk - 1 to 3 years" will include 31
      employees (2.4%)
   Allowing "Experience : XML - none" will include 78 employees (6.2%)
Edit?
```

This information is crucial to Stacy's understanding of the underlying staffing issues. Clearly, the company does not have many employees with the skill makeup of someone like Walter or Wendy. But more precisely, it is apparently the object-oriented skills, as opposed to the languages used, that are in such short supply. (She knows this because relaxing the object-oriented category constraint dramatically increased the size of the category, whereas relaxing the language constraints did not.) This tells Stacy that if workers like Walter are to be recruited or developed, the emphasis needs to be on object-oriented training rather than on the associated language training.

Stacy wonders if the lack of in-house training reported earlier from the four specific examples applies to the category company-wide. She leaves the category definition alone for now, and tells the system:

```
Find category correlations.
```

This tells the system to compare employees who fall under the current category definition with those who don't, and to report any statistical correlations it finds. The system computes for a while, and then reports:

```
Finding correlations in category "web services engineer" (31 employees)...
Employees in category "web services engineer" also tend to have:
   Job Satisfaction : previous year – medium to high  AND
   Seniority – 5 or less years  AND
   Age – 39 or less  AND
   { Project : Eclipse – 1 or more years  OR
     Project : EasyView – 1 or more years }  AND
   Training : in-house – none  AND
   { Career goal : Senior Developer  OR
     Career goal : Architect }
```

Stacy's suspicion is confirmed: there is apparently no in-house training course that tends to be common among employees with the desired skill set. This coupled with the fact that such employees tend to be fairly new to the organization suggests that they are receiving their object-oriented training from outside the company. A basic pattern is emerging that could have a radical impact on how upper management decides to manage the situation: workers with Wendy's skill set tend to be scarce, new, young, recruited from outside the company and not trained within.

Notice that these tools are helping Stacy get a picture of the characteristics of a particular type of worker. Nothing is black and white, of course, but since Stacy is operating at the strategic level she needs to think in generalizations and trends. Several possible action strategies are already becoming clear. She may investigate why in-house object-oriented training is not offered, or if it is, why it is not being taken by the relevant staff. She may survey managers of legacy products (i.e., not object-oriented products) and try to find out whether there is a general desire among their staff to acquire new skills and become proficient at the object-oriented paradigm. The question is: for this new style of programming, is it better to hire from outside, or to retrain part of the existing employee base? Clearly, for better or for worse, the company currently *is* adopting the former policy by default. However, this is almost certainly not the result of a well thought out, strategic decision, but rather the accumulation of many ad hoc, reactive, tactical decisions by first-level managers. The wisdom of this approach is a question Stacy intends to raise, and the database tools have given her the impetus and the information necessary to raise it.

Still more information can be found nestled within the system's brief response. Note that in examining the prototypical "web services engineer," the system has detected a correlation between skill set and career aspirations. Apparently, workers with skills in the requisite areas are typically interested in pursuing careers in development, rather than in other aspects of engineering, such as testing, maintenance, documentation, or management. This immediately raises a red flag in Stacy's mind. An object-oriented development project, like any other kind of project, will need to be handed off to a maintenance team once the code base is released, so that ongoing enhancement requests and bug fixes can be handled. Yet nearly everyone in the company with the training to perform such maintenance isn't interested in that line of work! There is potential for real disaster here, if object-oriented development projects truly are becoming more common as the managers she interviewed suggested; maintenance staff will simply be unable to support them. Perhaps some of the development staff can be convinced to migrate to maintenance activities, but the short seniority figure the system reported argues against this: workers in this category tend to fluctuate between jobs frequently, which suggests that many would likely seek opportunities outside the company rather than accept a shift in career direction. The system has revealed an impending avalanche which the maintenance management team might not fully appreciate, and it clearly needs to be brought to their attention.

Let's take a moment to review what the database has empowered Stacy to do thus far. She started with only a vague notion that there was a certain type of worker with a certain mix of skills, competencies, and interests that was perceived to be in high demand in the company. The tools have enabled her to:

- Formulate a precise description of what that prototypical worker's characteristics are.
- Estimate how many available resources in the company fit that description.
- Determine which of those characteristics are the most restrictive with respect to the employee pool.
- Learn what other characteristics those kinds of workers tend to have.
- Work at an abstract level. Note that Stacy did not need to approach the system with intimate knowledge about what constituted a particular worker type, or even

what the field types were. Instead, she was able to provide examples of employees already identified, and the system did the work to help her understand what characterizes them.

The standard relational database technologies of today do not facilitate *any* of the above functions. They can only be achieved by laboriously coaxing information out of the database by means of a large number of low-level queries that are awkward to pose.

4.2. Key concepts

The power of the system lies in its paradigm of *categories*. The basic units that the system operates on are the conceptual categories that a user identifies, and the examples of individual records that belong to them. From a small set of examples, the system can generalize to discover the salient attributes of the category they belong to, and form a more universal category definition that can be used to classify additional examples. This process is interactive; the system does not claim to have any deep knowledge about the real-world entities that the data records represent, but only reports the statistical correlations that it finds, which the user can reflect upon and adjust. Once the user and system jointly establish a category definition, more advanced operations upon that category become possible: broadening it, narrowing it, calculating its size, finding other correlations, comparing its relationship to other categories, finding additional instances, estimating "how closely" a particular instance conforms to its definition, monitoring and predicting its size over time, etc.

The category paradigm allows users to make sense of large data sets and to relate to that data at the abstract level they are accustomed to. Humans naturally form generalizations in order to function and to make sense of their world. No one gets into a new car and throws up their hands in frustration because they don't have any idea how to drive it. Even though that *particular* car may be unfamiliar to them, there are many attributes that all instances of the category "car" share; for instance, ignitions, steering wheels, accelerators and brake pedals. This knowledge allows a driver to operate nearly any vehicle simply by generalizing from their modest amount of experience.

The same is true with data. Knowing that there are 4,981 employees in a company isn't much help. And being able to access each of a hundred different fields for each of those employees doesn't help much either. The only way we can make sense of it all is to form generalizations about the kinds of employees in our staff, to partition the data in ways we can understand, and thus to recognize the macro-level patterns.

Every individual is unique, yet we know that there are a few dozen different *types* of workers in any organization. The employees in a particular category are not mindlessly interchangeable like machine parts, yet there are commonalities among them that we perceive, and that are useful in understanding the underlying issues. Different kinds of employees can handle different kinds of tasks and serve in different kinds of roles, and by understanding the trends and relationships between these different groups we can form a picture of what is really going on.

Allowing the user to operate at the level of *category* permits a far richer set of interactions. Instead of dealing with individual tables, rows, and fields, and being forced to manually reduce abstract questions to base elements and express them in awkward query languages, the user can treat the category itself as a bona fide object for investigation. This reflects what's in the user's mind, after all. The notion of exactly which records properly belong to a category conceptualization is an imprecise one, to be sure. But by giving regular feedback about exactly what criteria it's using, the user remains aware of how records are being classified and can fine-tune it if necessary. And the system is then free to provide the kinds of estimates and tendencies that the user is seeking at the most natural level.

4.3. Gaining insight over time

As she continues to use the system, Stacy will develop more and more categories in which to classify workers. Eventually there may be two dozen or more: *client-server designers* and *procedural programmers*, *unit test designers* and *integration testers*, *network administrators* and *infrastructure planners, domain experts* and *legacy system experts* for various extant product lines. The definitions of these categories evolve over time: as more and more workers are explicitly identified as examples of a particular

category, the system has more data to work with and forms a more precise understanding of what the qualifications of that category are. Some categories have overlap between them, and there is nothing to restrict a worker to belonging only to a single category. A few categories may correspond directly to existing job titles; most will not, representing instead a more complex mix of skills, experience, and interests that are applicable in a variety of settings.

Eventually, Stacy will develop a profound intuition about the makeup of the company's workforce as a whole, backed up by the system's quantitative data. And her view of what the data *is* will change. When she logs on to the database, she will no longer see it as being comprised of thousands of *records*, but rather as a much smaller number of *categories*, which can be examined and reasoned about in their own right. Although any individual employee record can still be retrieved in isolation, she will normally choose to view the database through the lens of abstraction, where the system communicates with her in terms that closely resemble her actual thought process.

The knowledge she can derive from the database is limited only by what kinds of data it contains and by her ability to combine and examine it. Here are some examples of things she might learn over time:

- In recent years, the number of employees satisfying the *internal technical writer* category has kept up with the overall company's rate of growth, but the number of such employees with an actual job title pertaining to "documentation" has not. This suggests that either (a) the burden of writing technical documents has been shifting from the actual documentation teams to other engineers, or (b) the documentation teams are now handling more work per writer, or (c) internal documentation is simply receiving less attention overall in the company. This issue could be further explored and possible actions taken.
- There appears to be a sufficient pool of resources in the *Eclipse technology trainer* category, which seems to imply that if the Eclipse product line is ramped up for new development as expected, training new employees in the proprietary design wouldn't be a problem. However, the system reports that this category has substantial overlap with the *network architect* category. Therefore, if new architecture duties will be taking place at the same time as the proposed training

phase, resources will in fact be limited because largely the same pool of employees is qualified for each. Either the two efforts need to be staggered in time, a different set of workers equipped as trainers, or some other evasive action taken to avoid this dilemma.

- Management is proposing the creation of a new research division, and will need a significant number of employees in the *wireless technology developer* category to staff it. The system reports that very few workers with the requisite skill set exist in the company. However, the category definition can be compared against the employee records from two different groups whose projects are scheduled to be phased out, and also against the records of new job applicants. The system computes that in general, the "delta" (difference) between the internal employees and the new position's required skills is smaller than that between the external applicants and those skills. This suggests that overall, the company would be better suited to transfer employees from within, rather than engaging in a lot of new hiring, to fill the required positions.

The categories themselves serve a number of other convenient functions. In addition to specifying the criteria for an employee to belong to them, they can quantitatively estimate "how close" any non-member is to satisfying them. This would be of great help to strategic staffers; for any particular open position, a pool of available employees (or new applicants) can be instantly compared with the relevant category definition. The results could be ranked according to how closely each of their records matched the criteria, and the areas of difference readily identified. A manager could see, "Joe is a close match, but it would be nice if he had these two training classes…Jill's skill set is perfect, but her career goals are not typical of what we usually see in this area, I might want to talk to her about that…"

Categories also serve to expose deficiencies in the data. There will be times when the managerial staff clearly identifies a particular category of worker, and can specify example employees who fit the intuitive understanding perfectly, but the system is unable to lock in on any reliable means of correlating them. In such cases, the user will get a message like:

```
Can't learn category definition...specify additional examples?
```

This alerts the user that some skill, aptitude, type of experience, or other data is not currently being collected, and probably should be. After all, managerial staff have recognized a valuable role or function, no trace of which is represented in the information the organization maintains. This is also a valuable clue that such contributions may have been going unnoticed, and perhaps also unrewarded.

4.4. Conclusion

Workforce planning may be an appropriate domain to make use of a tool for information space organization. Upper management needs to understand the patterns and trends underlying large amounts of personnel data in order to anticipate problems and craft a workforce suitable of carrying out the evolving mission of the organization. The proposed system may allow strategic staffers to relate to the data in a natural way, enhancing their own understanding and decision-making process.

5. Research plan

My research and development plan consists of five phases, each of which is described in detail below.

5.1. Phase one - Preparation

In this phase, I will perform the tasks necessary to put me in a position to actually build a useful, non-trivial software product. The most important goal is to find a domain (or domains) to which I can apply my methodology, and experts from that domain who are willing to work with me. In parallel with this, however, I will also perform some domain-independent technical work that will be necessary to support the platform.

5.1.1. Domain selection

I desire to apply my methodology to a real-world setting in order to determine its utility. If successful, this will make the ideas as believable as possible, and it should also

expose any fundamental weaknesses in the approach and suggest ways to correct them. I will seek out domains which feature the following criteria:

1. A need for information space organization. Representatives from the domain must perceive the need for a deep, universal, lasting understanding of their entire information space. It would be a mistake to try to apply my tool to a domain in which the professionals are perfectly content with obtaining isolated facts via the query-response model, and can see no reason for a systemization of the entire space.
2. Appropriate size. The domain must involve enough data that a tool like mine is needed, but not so much that it is hopeless. For example, a school teacher might need to categorize her class of thirty students so that she knows which teaching methodologies are the most appropriate, and which students may need special attention. However, a data set this small would not demonstrate the power of the tool – it would probably be just as easy to form a categorization by hand. Conversely, any domain that claims "the Web" as its information space would be doomed in its attempts to gain a comprehensive organization of it – the sheer magnitude makes this impossible. Therefore, I seek domains in which the data is voluminous enough that no human can categorize it unaided, yet manageable enough that automated tools have a chance.
3. Structured data. The data for the domain must include coded, unambiguous information that can be represented in a Semantic Web format and operated on by the machine learning algorithms I envision. Purely natural language text or multimedia data is not ideal for my system.
4. Complex relationships. The data must be complex enough that meaningful patterns can be identified. A simple list of customer names and addresses, for instance, is unlikely to lead to many profound categorizations.

I have identified a number of candidate domains that meet these general criteria. Table 2, below, lists several of them, and identifies how each one meets these criteria.

Possibly applicable domain	*Satisfaction of key criteria*
Personnel management. Workforce planners proactively manage their staff (as per the scenario in section 4.)	Need: Managers must perceive trends in the workforce, not merely individual employee attributes. Size: Hundreds or thousands of workers. Structure: Skill sets, achievement scores, career goals, etc. Complexity: A given position may require a mix of skills and attributes, for instance.
Literature review. A researcher wants to learn the lay of the land in a particular field.	Need: Researchers must know the general direction of the field, which areas are heavily researched and where there are gaps, who is working with whom at what institutions, what the important publications are, etc. Size: Thousands of publications. Structure: Documents are annotated with author, date, institution, conference/journal, topic keywords, references to other papers. Complexity: The researcher may identify groups of publications that contain related key ideas, for instance.
Public health. Health officials need to track trends within populations, identify correlations between symptoms and various possible causes, and distinguish between cases of true epidemics like SARS from those that might present similar features (such as the common flu.)	Need: A global view of a population is required in order to manage epidemics. Isolated incidents alone will not yield a complete picture. Size: Tens of thousands of reported cases, implicitly representing a population in the millions. Structure: Coded information as to differential diagnosis, chief complaint, symptom profile, age, gender, occupation, geographic location, etc. Complexity: Dangerous diseases may spread through a variety of channels, and may manifest themselves in a complex array of symptoms that can be difficult to distinguish from more common ailments.
Real estate. Agents need to categorize properties and their clients so that they can make precise recommendations, predict which way the market is turning, know which types of homes are saturated with buyers and which might be good deals, and so on.	Need: Agents deal in a house at a time, but must remain acutely aware of the trends in the general market to stay competitive. Size: Thousands of properties for sale. Structure: Detailed property descriptions include asking price, property taxes, square footage, types of rooms, location, amenities, and literally hundreds of other items. Complexity: Buyers can be categorized into various groups, each of which tends to seek a certain mix of criteria in a home.
Life sciences. Researchers in biology and zoology compile voluminous data on the characteristics and habitats of organisms.	Need: Scientists look for patterns and trends within large habitats, try to understand population drifts, identify common themes and relationships between species. Size: Perhaps five million distinct species on earth. Structure: Specialists in various areas have rigorously identified the structural, behavioral, and genetic characteristics that differentiate species from one another. Complexity: Species grouped according to key properties; habitats compared with each other to identify cause-effect relationships.
Finance. Professional investors survey investment options and make decisions about how to meet short-term and long-term financial goals.	Need: Financial planners need a broad view of what types of financial instruments are available, and what tendencies in these groups emerge over time, in order to advise their clients. Size: Thousands of different stocks, bonds, certificates, and funds. Structure: Investments are labeled by broad category, past performance, key holdings, earnings ratios, etc. Complexity: A balanced portfolio must emphasize various facets of the market to varying degrees, depending on the stated financial goals of the investor.

Possibly applicable domain	*Satisfaction of key criteria*
Culinary arts. Professional chefs and caterers need to get a handle on their recipe base in order to appropriately plan meals, satisfy their clientele, and minimize costs.	Need: Different types of meals are appropriate for different venues, and have different cost profiles based on the number served. Many caterers serve conventions or other multi-day events, and require sequences of meals that offer variety and balance of tastes, while at the same time meeting competitive budgets. Size: Hundreds of dishes that can be mixed and matched in various ways. Structure: Ingredients, preparation type, preparation time, portion sizes, cost, serving requirements, ambiance, suitability for particular seasons or holidays, popular combinations with other dishes, etc. Complexity: Designing a successful meal package is both an art and a science. It requires a combination of many different factors to please clientele, meet budget, and procure future business.
Marketing. Sales divisions need to perceive trends in their customer base, target advertisements to particular groups, understand and parameterize buying patterns.	Need: Individual customers are not in view; it is rather groups of customers and what buying patterns can be expected of them. Advertisements are costly to deliver, and it is highly advantageous to target them to the right consumers. Size: Widely varying, depending on the size of the company. Structure: Demographics information, records of previous purchases, registration surveys. Complexity: The target market for a particular product may be defined by a combination of factors: location of residence, occupation, homeowner/renter, income level, hobbies/interests.
Voting demographics. A campaign manager needs to identify key groups of people within a population in order to recognize which issues they perceive as critical, what their voting tendencies are, and how to target custom advertisements.	Need: Potential voters can be categorized into a number of important groups, some of which may be of special interest in a particular election. Size: Thousands of respondents to surveys, implicitly representing a population in the millions. Structure: Demographics information, responses to public opinion polls that reveal values and important issues, approval ratings, likelihood of voting, etc. Complexity: A key constituency – such as the recently publicized "soccer moms" and "NASCAR dads" phenomena – may require a number of different factors to define, and may reveal an array of correlations that could impact advertising decisions.
Legal. Attorneys need a comprehensive understanding of certain areas of case law, so that they can cite the most appropriate cases and anticipate what their opponents may cite.	Need: An attorney must cite individual cases, but the bewildering array of past litigation needs to be organized so that those individual cases can be readily found. Size: Thousands of potentially applicable cases in an area of law. Structure: Each case can be parameterized by the demographics of the participants, presiding judge, relevant statutes, damages sought and awarded, compelling factors, citations in other cases, etc. Complexity: What makes a given case relevant to another varies from situation to situation and depends on the strategies involved.

Table 2. A sampling of domains for which an information space organization tool might be useful. The right column relates each domain to the four key criteria identified above.

It seems to me clear that prospective domains abound; it is a matter of finding experts in an area who can understand the principles I have identified and who are willing to work with me on building an experimental prototype. Thus far, I have made contacts in personnel management (State of Colorado Human Resources), public health (Denver Public Health), and life sciences (the Smithsonian Encyclopedia of Life project) who are very enthusiastic about working with me. In this preparatory phase, I will continue to work closely with these domain representatives, as well as seeking additional contacts from other areas. Based on the criteria listed above, the needs of the participants, and the quality of the available data, I will choose one or two domains for which to customize my system. My goal is to work with these domain experts throughout the project in order to constantly relate my technical advancements to the pragmatics of everyday decision-making.

5.1.2. Assemble infrastructure

In parallel with selecting a domain, I will begin to build the domain-independent parts of the system architecture. This will include investigating the freely available Semantic Web tools listed in section 2.2.6, and choosing an RDF store, query interface, and server-side architectural components. I will know the relevant products and their tradeoffs by the end of this phase, and will have made appropriate selections.

5.1.3. Design and implement web crawler component

My architecture calls for a daemon process that will continually scan network domains of interest and incorporate the semantic annotations that it finds into the RDF store so that they can be processed quickly. This component should be largely domain-independent, and will be necessary for even basic functionality once real data is assimilated. I will build this component in the preparation phase so it is ready in advance.

I will emerge from this phase having chosen a specific domain, recruited partners who are committed to working with me, and established some baseline architecture on which I can construct a real system.

5.2. Phase two - Interface design

This phase will demand close work with domain representatives. I will thoroughly interview experts in order to understand the business processes at work, the way the data is perceived, and the key questions that need to be answered in that domain. In the spirit of task-centered design, I will develop a user interface model that will benefit those experts' particular functions. At the same time, however, I will always look to the general case to ensure that most major aspects of the model are transferable outside that particular domain area.

There is a delicate balance at work here. On the one hand, I want to appreciate the way a particular group of experts currently does business, so that I can develop a tool that satisfies their real, present needs. On the other hand, a tool like mine may enable functions that these experts have never before envisioned. Therefore, I don't want to merely take their existing processes and codify them into a system, if in reality greater benefit would arise by making available an entirely new paradigm. Hence I will constantly be considering two factors – what types of behavior do experts in the domain immediately recognize as useful, and what behaviors are outside their current frame of reference yet may prove to be a better way of doing business.

It will also be important to consider evaluation methods in this phase. By discussing the problems that the domain workers regularly face, it should be possible to arrive at techniques for evaluating how quickly and how well they can be solved. I will work with the domain experts to quantify significant metrics that can be tested later, and also to glean what sort of subjective assessments are appropriate. (See also section 5.5, below.)

I will emerge from this phase with a well-documented first attempt at a system interface. I will also have a list of problems that need to be solved in that domain, concrete examples, and well-developed ideas of how solutions can be tested both objectively and subjectively.

5.3. Phase three - Implementation

Based on the insights gleaned from domain experts, I will construct a prototype system for information space organization. It will be tailored to the domain in question, and may even provide certain features that are nontransferable, but the central functionality will be broadly applicable. The tasks in this phase correspond to the main components in the system architecture, as depicted in Figure 2.

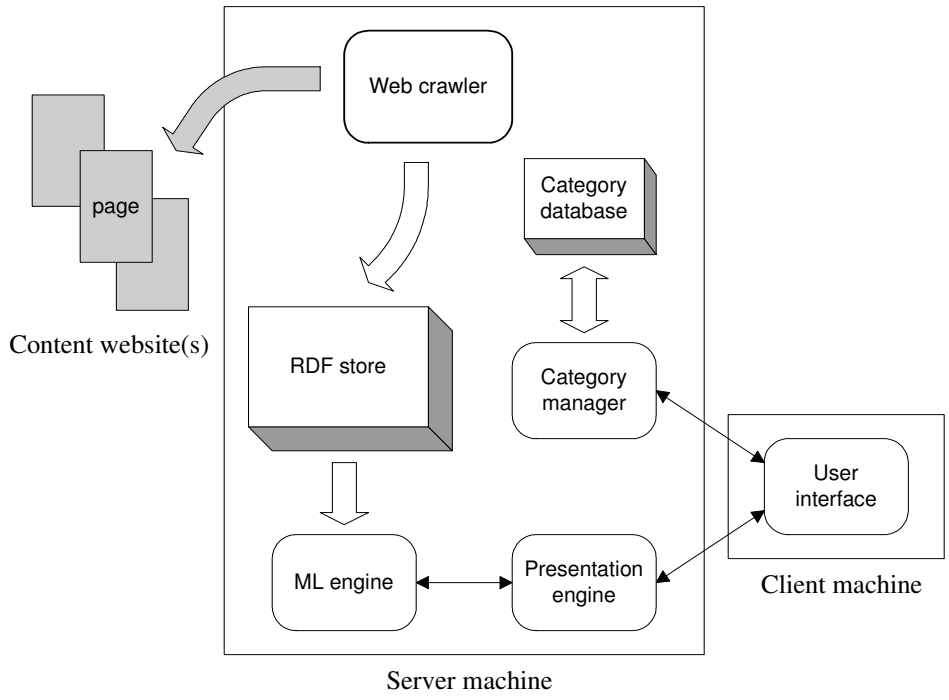

Figure 2. Preliminary architecture diagram. A web crawler component operates in the background, searching prescribed network sites for pages with RDF annotations, and recording this in the optimized RDF store. Other server components cooperate to (1) analyze instances; (2) create, store, and reason about categories; and (3) direct the user experience by selecting instances to display. The user interface component runs on a client machine and allows the user to view and manipulate constructs such as instances and categories. Depending on the interface requirements, the client may be thin (e.g., SVG code rendered in a browser) or thick (e.g., standalone Java Swing application.)

5.3.1. User interface

The feedback from the previous phase will go a long way towards determining what technology I choose for the user interface. My preference would be to implement a thin client based on the scalable vector graphics (SVG) protocol[46], as this is

lightweight and extremely portable. I may discover, however, that the features required by workers in the domain demand richer, more responsive GUI components than can be delivered by such a scheme. In this case, I will develop a standalone client, probably written with Java Swing for portability purposes, that communicates to the server via the Web Services protocol[21].

In either event, the major functions of this interface will be as described in section 3 and illustrated in section 4, above.

5.3.2. Web crawler

As described in phase one, a Web crawler component will asynchronously comb a prescribed set of networks looking for pages with RDF annotations, and store them in an open source database specifically designed to house RDF triples. My hope is that by the time I reach phase three, this is already operational, and the other components can operate solely on the compiled information.

5.3.3. RDF store

Many RDF databases have been designed by the Semantic Web community and are freely available. My initial preference is to use Dave Beckett's Redland framework[15], because of its popularity, reputation for robustness, and multi-language interface suite.

5.3.4. Machine learning engine

This is the most performance-intensive component of the system. It is responsible for providing numerical answers to questions about category-level constructs, which it understands. For instance, given an instance and a category, it can compute the likelihood of that instance belonging to the category. It can estimate the size of a category (in number of instances), the degree of overlap between two categories, the statistically relevant attributes of a category or intersection of categories, and so on. Also, by examining a category's examples and counterexamples, the engine can form a hypothesis about the category's "true" definition, and a confidence measure in that

hypothesis. Thus it provides the major computational building blocks that the other system components need in order to deliver an information space organization experience. The machine learning engine interrogates the RDF store directly to derive its results.

5.3.5. Category manager

The category manager stores a record of the semantics each user has assigned to the information space, in the form of persistent categories. Categories can be retrieved, analyzed, manipulated, and shared between users. Retrieving and saving categories is not expected to be a time-critical operation, so I will probably implement this component in terms of flat files stored on the server. If this proves to be unsuitable, I will use MySQL or another open source relational database to house category information.

5.3.6. Presentation engine

The presentation engine directs the user interface experience by deciding which instance to show the user after each interaction. This is based on a combination of factors, as described in section 3.1, above.

5.3.7. Content pages

Finally, it is likely that I will need to convert the data that my domain partners provide into Semantic Web format in this phase. The data set I seek will not be overwhelmingly large (only as large as necessary to demonstrate my research ideas), and I expect that I will be able to write some simple automated tools to put this together. The result will be a set of semantically annotated web pages, human-viewable and machine-readable, that will be deployed on some known server or group of servers. The web crawler component will be configured to know the addresses of these servers so that it can find the material it needs to work on.

5.4. Phase four – Deployment

After I have constructed this initial prototype and tested it to my satisfaction, I will deploy it to my domain partners onsite for their experimentation. The result will be an iterative process, where both parties jointly discover flaws in the interface, missing features, and other practical challenges. We will apply the tool specifically to the domain problems identified in phase two, and assess how well it contributes to a solution. My intention is to continue this process of iteration until the major issues have been identified and the tool is capable of being applied in a real setting. I do not intend to achieve commercial-grade robustness, or to implement an endless array of domain-specific features. Once the tool's major functionality is operable and applicable, that is enough.

5.5. Phase five - Evaluation

The tool will be deployed so that real domain experts can experiment with it to perform real tasks. Depending on the domain, I may or may not attempt to arrange an actual beta test on live data. This will depend on how large the raw data set is (whether or not it is feasible to assimilate it all given the equipment at my disposal), and the preferences of my domain partners. If I elect to pursue the workforce development scenario, for instance, I may be able to assimilate an entire set of current data, if the organization is of a modest size. Other domains (say, legal) may call for a trimmed down "analysis data set" instead so that users can experiment with the system on data that is truly reflective yet constrained. The scalability issue is one that ultimately needs to be addressed, of course, if tools like the one proposed are to be effective in the real world. Since this is not the focus of my study, however, I will defer such issues to a future investigation, assuming that the main user interaction paradigm proves effective.

Evaluation of the tool will be twofold: subjective, and objective. Subjectively, I will work with my domain partners to construct a list of specific questions appropriate to the analysis of the tool in that domain setting. Experimental users will be directed to consider these questions as they work with the system, and provide anecdotal answers. These will pertain to how well the tool's functionality tracks with the information needs

of that domain, the level to which users feel they can understand the information space as a whole, and ease of use. It is difficult to be more specific than this, since the ways in which users will want to organize an information space depend very much on the particular domain. Detailed interviews will therefore be necessary in order for me to ascertain exactly what kinds of questions a representative of the domain will want to ask, and what kinds of categories (or other organizational constructs) are necessary for them to feel they have an intuitive understanding of the space that can impact future decisions.

Objectively, I hope to implement something like Pirolli's and Card's information gain methodology[93-95]. By measuring the rate at which "relevant" instances (as judged by the user) are encountered, I will aim to devise a controlled study where the system can be compared with an ordinary Boolean search engine. This should give some kind of quantitative measure of how well the system focuses the user experience on gaining practical knowledge about the nature of the information space.

My belief is that these two methods combined will give an idea of the extent to which the proposed information space organization tool is helpful and practical. It should also reveal shortcomings in the paradigm, both domain-specific and general, and perhaps suggest practical ways to address them. Admittedly, since the project involves a completely new interaction model, it is difficult to know at the outset exactly how best to assess it – its benefits are rather intangible and cannot be compared side-by-side with those of existing systems in a straightforward way. Indeed, I expect that one of my contributions will be in discovering how best to *measure* how well an information space has been organized, in addition to developing tools to actually perform the task. All of these findings will be documented and will form an important part of the contribution I plan to make with this dissertation.

6. Conclusion

Information space organization is a real need that has gone largely unnoticed in the human-computer interaction community. In order to accomplish it, people today depend on the power of their own fallible memories and on tools that are ill-suited to the task. It is possible, however, to develop applications specifically to aid the organization

process, and that make use of existing data mining algorithms to help identify the structure more quickly.

The machine-readable annotations of the Semantic Web were not conceived with this purpose directly in mind, but they offer a tremendous opportunity for leveraging this kind of technology. I have delineated some desired principles for an information space organization tool, and sketched a preliminary architecture that could achieve them for Semantic Web data. My plan is to implement such a system for an actual real-world domain and evaluate how well it can enhance human understanding of large data sets.

7. Appendix – Semantic Web Technologies

This document has made reference to a number of basic Semantic Web concepts relevant to my work. These include URIs and URLs[17], XML[22], and the RDF language[14].

- A URI (uniform resource identifier) is the most general kind of "label" that can denote a particular entity. URIs are unique, unambiguous identifiers that refer to any kind of instance about which assertions can be made: these can be instances of web pages, documents, or real-world entities like corporations or people.
- A URL (uniform resource locator) is a special kind of URI: one that represents an addressable location in cyberspace.
- XML is an enormously popular data representation syntax that is becoming ubiquitous in Internet-based data interchange. It is both human-readable and machine-processible, which gives it wide applicability. XML allows custom tags with no preset semantics. Each instance of an XML document has a single root element that contains optional attributes and also a hierarchy of subelements. Hence XML is by nature hierarchical. Its custom tags allow it to represent nearly any form of structured data unambiguously.
- The RDF (Resource Description Framework) language is currently the most popular way of expressing logical assertions about resources. Each RDF statement is a "triple" that contains a subject, predicate, and object, all three of which are specified with URIs rather than free text.[10] In this way, a large body of RDF statements can be seen as representing a graph: the subjects and objects are nodes, and the predicates are (labeled) edges. This RDF graph, embedded into

[10] An example might be:

```
http://blah.com/aDocument  http://purl.org/dc/elements/1.1/wasAuthoredBy
   http://blah.com/employees/1798
```

which asserts that the given document was written by employee #1798 at Blah, inc. Note that the predicate of this statement ("wasAuthoredBy") is a URI referring unambiguously to an element of a standard ontology, just as the subject and object are unambiguous URIs. This makes everything explicit, allowing a reasoning engine to incorporate this statement and draw conclusions from it with no uncertainty. Note also that the URIs are not necessarily URLs: for instance, if one typed "http://blah.com/employees/1798" into one's browser, it would almost certainly result in "page not found." This is because that URI is not actually a web-addressable resource – it is simply a unique string that refers semantically to a real-world entity, namely, a particular person.

web pages in the form of individual assertions[11], forms the warp and the weft of the Semantic Web. It is the very structure I intend my system to analyze.

[11] At the lowest level of detail, RDF statements are often serialized into an XML-compatible format for embedding within web pages. This common method of representation is chosen for merely pragmatic reasons; there is no reason why RDF triples cannot be expressed in other ways, and in fact there are other less common serialization standards available, such as N3 notation. At any rate, the important point is that conceptually, the RDF assertions comprise a graph, and the fact that XML is used to encode that graph is a detail.

8. References

1. *Grokker 2.1*. Sausalito, California, 2004. Available at: www.grokker.com.
2. *The World Wide Web Consortium*. 2004. Available at: www.w3c.org.
3. *XML TopicMaps (XTM)*. 2001. Available at: http://www.topicmaps.org/xtm/index.html.
4. Aberer, K., P. Cudre-Mauroux, and M. Hauswirth. *The Chatty Web: Emergent semantics through gossiping*. in *Proceedings of the 12th International World Wide Web Conference*. 2002. Budapest, Hungary.
5. Agrawal, R., T. Imielinski, and A. Swami. *Mining association rules between sets of items in large databases*. in *Proceedings of the ACM SIGMOD Conference on Management of Data*. 1993. Washington, D.C.
6. Ahlberg, C., C. Williamson, and B. Shneiderman. *Dynamic queries for information exploration: An implementation and evaluation*. in *Proceedings of the ACM Conference on Human Factors in Computer Systems*. 1992. Monterey, California.
7. Alexaki, S., V. Christophides, G. Karvounarakis, and D. Plexousakis*, The ICS-FORTH RDFSuite: Managing voluminous RDF description bases*. Institute of Computer Science, FORTH: Heraklion, Greece, 2001. Available at: http://www.ics.forth.gr/isl/publications/paperlink/semweb2001.pdf.
8. Ankerst, M., M. Ester, and H.-P. Kriegel. *Towards an effective cooperation of the user and the computer for classification*. in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2000. Boston, Massachusetts.
9. Ankolekar, A., M. Burstein, J.R. Hobbs, O. Lassila, D. Martin, D. McDermott, S.A. McIlraith, S. Narayanan, M. Paolucci, T. Payne, and K. Sycara. *DAML-S: Web Service description for the Semantic Web*. in *First International Semantic Web Conference (ISWC 2002)*. 2002. Sardinia, Italy.
10. Arens, Y., C.A. Knoblock, and W.-M. Shen, *Query Reformulation for Dynamic Information Integration.* Journal of Intelligent Information Systems, 1996. **6**: p. 99-130.
11. Assini, P. *NESSTAR: A Semantic Web application for statistical data and metadata*. in *Eleventh International World Wide Web Conference: Workshop on Real World Applications of RDF and the Semantic Web*. 2002. Honolulu, Hawaii.
12. Baleydier, L.*, The KartOO Visual Metasearch Engine*. Clermont-Ferrand, France, 2004. Available at: www.kartoo.com.
13. Bayardo Jr., R.J., W. Bohrer, R. Brice, A. Cichocki, G. Fowler, S. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. *Infosleuth: Semantic integration of information in open and dynamic environments*. in *Proceedings of the 1997 ACM-SIGMOD International Conference on Management of Data*. 1997. Tuscon, Arizona: Morgan Kaufman.
14. Beckett, D.*, RDF/XML Syntax Specification (Revised)*. World Wide Web Consortium: 2004. Available at: http://www.w3c.org/RDF/.

15. Beckett, D., *Redland RDF Application Framework*. University of Bristol, UK: 2000. Available at: http://www.redland.opensource.ac.uk.
16. Berendt, B., A. Hotho, and G. Stumme. *Towards semantic web mining*. in *Proceedings of the First International Semantic Web Conference*. 2002. Sardinia, Italy.
17. Berners-Lee, T., R. Fielding, and L. Masinter, *Uniform Resource Identifiers (URI): Generic Syntax*. Network Working Group: Request for Comment 2396: 1998. Available at: http://www.ietf.org/rfc/rfc2396.txt.
18. Berners-Lee, T., J. Hendler, and O. Lassila, "The Semantic Web," in *Scientific American*, May 2001, 2001.
19. Bertino, E., B. Catania, and G.P. Zarri, *Intelligent Database Systems*. 2001, Edinburgh, England: Addison-Wesley.
20. Boley, D., M. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore, *Document categorization and query generation on the World Wide Web using WebACE.* AI Review, 1999. **13**(5-6): p. 365-391.
21. Booth, D., H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, *Web Services Architecture*. World Wide Web Consortium: 2004. Available at: http://www.w3c.org/TR/ws-arch/.
22. Bray, T., J. Paoli, C.M. Sperberg-McQueen, and E. Maler, *Extensible Markup Language (XML) 1.0 (Second Edition)*. World Wide Web Consortium: 2000. Available at: http://www.w3.org/TR/2000/REC-xml-20001006.html.
23. Brickley, D. and R.V. Guha, *Resource Description Framework (RDF) Schema Specification 1.0*. World Wide Web Consortium: 2000. Available at: http://www.w3.org/TR/2000/CR-rdf-schema-20000327/.
24. Broekstra, J., A. Kampman, and F. van Harmelen. *Sesame: A generic architecture for storing and querying RDF and RDF schema*. in *First International Semantic Web Conference*. 2002. Sardinia Italy: Springer-Verlag Heidelberg.
25. Carenini, G. *An analysis of the influence of need for cognition on dynamic queries usage*. in *Proceedings of the Conference on Human Factors in Computing Systems*. 2001. Seattle, Washington.
26. Chakrabarti, S. and Y. Batterywala. *Mining themes from bookmarks*. in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2000. Boston, Massachusetts.
27. Chawathe, S., H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. *The TSIMMIS Project: Integration of Heterogeneous Information Sources*. in *16th Meeting of the Information Processing Society of Japan*. 1994. Tokyo, Japan.
28. Chen, Q., X. Wu, and X. Zhu. *OIDM: Online interactive data mining*. in *Proceedings of the 17th International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems*. 2004. Ottowa, Canada.
29. Chuang, T.-T. and S.B. Yadav, *A Conceptual Model of an Adaptive Decision Support System (ADSS)*. Texas Tech University: Lubbock, Texas, 1996. Available at: http://hsb.baylor.edu/ramsower/ais.ac.96/papers/chuang.htm.
30. Cohn, D., "Learning the structure of unstructured document bases." Microsoft Research: Seattle, Washington. 2001. Available at

http://murl.microsoft.com/videos/msr/MSR2001/Cohn_Learning_OnDemand_100_100k_320x240Slides.htm.
31. Cooley, R., B. Mobasher, and J. Srivastava. *Web mining: Information and pattern discovery on the World Wide Web*. in *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence*. 1997.
32. Craven, M., D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and C.Y. Quek. *Learning to extract symbolic knowledge from the World Wide Web*. in *Proceedings of the 15th National Conference on Artificial Intelligence*. 1998. Madison, Wisconsin.
33. Daconta, M.C., L.J. Orbrst, and K.T. Smith, *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. 2003: John Wiley & Sons. 312.
34. Davies, J., A. Duke, and A. Stonkus. *OntoShare: Using ontologies for knowledge sharing*. in *WWW2002 Semantic Web workshop, Eleventh International World Wide Web Conference*. 2002. Honolulu, Hawaii.
35. Davies, J., D. Fensel, and F. van Harmelen, eds. *Towards the Semantic Web: Ontology-Driven Knowledge Management*. 2003, John Wiley & Sons, Ltd.
36. Davies, J., U. Krohn, and R. Weeks. *QuizRDF: search technology for the semantic web*. in *WWW2002 workshop on real world RDF & Semantic Web Applications, 11th International WWW Conference*. 2002. Hawaii, USA.
37. Deerwester, S., S.T. Dumais, G.W. Furnas, T. Landauer, and R. Harshman, *Indexing by latent semantic analysis.* Journal of the American Society for Information Science, 1990. **41**(6): p. 391-407.
38. Engels, R.H.P. and T.C. Lech, *Generating ontologies for the Semantic Web: OntoBuilder*, in *Towards the Semantic Web: Ontology-Driven Knowledge Management*, J. Davies, D. Fensel, and F. van Harmelen, Editors. 2003, John Wiley & Sons, Ltd.
39. Erdmann, M. and R. Studer, *How to structure and access XML documents with ontologies.* Data Knowledge Engineering, 2000. **36**: p. 317-335.
40. Fails, J.A. and D.R. Olsen. *Interactive machine learning*. in *Proceedings of the Eighth International Conference on Intelligent User Interfaces*. 2003. Miami, Florida.
41. Farquhar, A., R. Fikes, and J. Rice, *The Ontolingua Server: a tool for collaborative ontology construction.* International Journal of Human-Computer Studies, 1997. **46**(6): p. 707-727.
42. Fensel, D. and C. Bussler*, The Web Service Modeling Framework WSMF Extended abstract*. Vrije Universiteit Amsterdam: 2002. Available at: http://informatik.uibk.ac.at/~c70385/wese/wsmf.bis2002.pdf.
43. Fensel, D., J. Hendler, H. Lieberman, and W. Wahlster, eds. *Spinning the Semantic Web*. 2003, The MIT Press: Cambridge, Massachusetts.
44. Fensel, D., I. Horrocks, F. van Harmelen, D. McGuinness, and P.F. Patel-Schneider, *OIL: Ontology infrastructure to enable the Semantic Web.* IEEE Intelligent Systems, 2001. **16**(2): p. 38-45.
45. Fensel, D., S. Staab, R. Studer, F. van Harmelen, and J. Davies, *A future perspective: Exploiting peer-to-peer and the Semantic Web for knowledge management*, in *Towards the Semantic Web: Ontology-Driven Knowledge*

Management, J. Davies, D. Fensel, and F. van Harmelen, Editors. 2003, John Wiley & Sons, Ltd.
46. Ferraiolo, J., F. Jun, and D. Jackson*, Scalable Vector Graphics (SVG) 1.1 Specification*. World Wide Web Consortium: 2003. Available at: http://www.w3.org/TR/SVG11/.
47. Fluit, C., H. ter Horst, J. van der Meer, M. Sabou, and P. Mika, *Spectacle*, in *Towards the Semantic Web: Ontology-Driven Knowledge Management*, J. Davies, D. Fensel, and F. van Harmelen, Editors. 2003, John Wiley & Sons, Ltd.
48. Frye, C., M. Plusch, and H. Lieberman, *Static and Dynamic Semantics of the Web*, in *Spinning the Semantic Web*, D. Fensel, et al., Editors. 2003, The MIT Press: Cambridge, Massachusetts.
49. Furnas, G.W. and S.J. Rauch. *Considerations for information environments and the NaviQue workspace*. in *Proceedings of the ACM Conference on Digital Libraries*. 1998: ACM.
50. Getoor, L., N. Friedman, D. Koller, and A. Pfeffer, *Learning probabilistic relational models*, in *Relational Data Mining*, S. Dzeroski and N. Lavrac, Editors. 2001, Springer-Verlag.
51. Ghani, R., R. Jones, D. Mladenic, K. Nigam, and S. Slattery. *Data mining on symbolic knowledge extracted from the Web*. in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2000. Boston, Massachusetts.
52. Gil, Y. and V. Ratnakar. *TRELLIS: An interactive tool for capturing information analysis and decision making*. in *Proceedings of Ontologies and the Semantic Web: 13th International Conference*. 2002. Siguenza, Spain.
53. Grosky, W.I., D.V. Sreenath, and F. Fotouhi, *Emergent semantics and the multimedia semantic web.* SIGMOD Record, 2002. **31**(4): p. 54-58.
54. Gruber, T., *Ontolingua: A mechanism to support portable ontologies.* Knowledge Systems Lab, Stanford University, 1992.
55. Gudgin, M., M. Hadley, N. Mendelsohn, J.-J. Moreau, and H.F. Nielson*, SOAP Version 1.2*. World Wide Web Consortium: 2003. Available at: http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.
56. Hastie, T., R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. 2001, New York: Springer-Verlag.
57. Hearst, M.A., A.Y. Levy, C.A. Knoblock, S. Minton, and W. Cohen, *Information integration.* IEEE Intelligent Systems, 1998. **13**(5): p. 12-24.
58. Heflin, J. and J. Hendler. *Searching the Web with SHOE*. in *Artificial Intelligence for Web Search. Papers from the AAAI Workshop*. 2000. Menlo Park, CA: AAAI/MIT Press.
59. Horrocks, I. and P.F. Patel-Schneider, *Optimizing Description Logic Subsumption.* Journal of Logic and Computation, 1999. **9**(3): p. 267-293.
60. Iosif, V., P. Mika, R. Larsson, and H. Akkermans, *Field experimenting with Semantic Web tools in a virtual organization*, in *Towards the Semantic Web: Ontology-Driven Knowledge Management*, J. Davies, D. Fensel, and F. van Harmelen, Editors. 2003, John Wiley & Sons, Ltd.

61. Jain, R., *Emergent semantics and experiential computing.* University of California, San Diego and PRAJA inc., 2000.
62. Jasper, R. and M. Uschold, *Enabling Task-Centered Knowledge Support through Semantic Markup*, in *Spinning the Semantic Web*, D. Fensel, et al., Editors. 2003, The MIT Press: Cambridge, Massachusetts.
63. Karvounarakis, G., V. Christophides, and D. Plexousakis, *Querying Semistructured (Meta) Data and Schemas on the Web: The case of RDF & RDFS.* Technical Report 269, ICS-FORTH, 2000.
64. Kifer, M., G. Lausen, and J. Wu, *Logical foundations of object-oriented and frame-based languages.* Journal of the ACM, 1995. **42**(4): p. 741-843.
65. Kiryakov, A. and D. Ognyanov. *Tracking changes in RDF(S) repositories*. in *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management*. 2002. Siguenza, Spain.
66. Kushmerick, N., *Wrapper induction: Efficiency and expressiveness.* Artificial Intelligence, 2000. **118**(1-2): p. 15-68.
67. Landauer, T. and S.T. Dumais, *A solution to Plato's problem: The Latent Semantic Analysis theory of acquisition, induction, and representation of knowledge.* Psychological Review, 1997. **104**(2): p. 211-240.
68. Lassila, O. and R.R. Swick*, Resource Description Framework (RDF) Model and Syntax Specification*. World Wide Web Consortium: 1999. Available at: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.
69. Le Grande, B., M. Soto, and D. Dodds. *XML Topic maps and semantic web mining*. in *XML Conference and Exposition 2001*. 2001. Orlando, Florida: IdeAlliance.
70. Lenat, D.B. and R.V. Guha, *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. 1990, Boston: Addison-Wesley.
71. Levy, A.Y., A. Rajaraman, and J.J. Ordille. *Querying heterogeneous information sources using source descriptions*. in *Proceedings of 22th International Conference on Very Large Databases*. 1996. Mumbai, India: Morgan Kaufmann.
72. Lewis, P.M., A. Bernstein, and M. Kifer, *Databases and Transaction Processing: An Application-oriented Approach*. 2002: Addison-Wesley.
73. Lieberman, H., *Personal assistants for the web: an MIT perspective*, in *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet*, M. Klusch, Editor. 1999, Springer-Verlag: Berlin. p. 279-292.
74. Lieberman, H. and T. Selker, *Out of context: computer systems that adapt to, and learn from context.* IBM Systems Journal, 2000. **39**(3): p. 617-631.
75. Linden, G., B. Smith, and J. York*, Amazon.com recommendations: item-to-item collaborative filtering*. IEEE Distributed Systems Online: 2003. Available at: http://dsonline.computer.org/0301/d/w1lind.htm.
76. Liu, B., C.W. Chin, and H.T. Ng. *Mining topic-specific concepts and definitions on the web*. in *Proceedings of the 12th International World Wide Web Conference*. 2003. Budapest, Hungary: ACM.
77. Luke, S. and J. Heflin*, SHOE 1.01 (proposed specification)*. 2000. Available at: http://www.cs.umd.edu/projects/plus/SHOE/spec.html.
78. Lyman, P. and H.R. Varian*, How Much Information*. University of California at Berkeley School of Information Management and Systems: Berkeley, California,

2003. Available at: http://www.sims.berkeley.edu/research/projects/how-much-info-2003/.
79. Maedche, A. and S. Staab. *Discovering conceptual relations from text*. in *Proceedings of the 14th European Conference on Artificial Intelligence*. 2000. Berlin: IOS Press.
80. Maedche, A. and S. Staab. *Measuring similarity between ontologies*. in *Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management*. 2002. Madrid, Spain.
81. Maedche, A. and S. Staab. *Mining Ontologies from Text*. in *Knowledge Acquisition, Modeling, and Management: Proceedings of the European Knowledge Acquisition Conference*. 2000. Berlin: Springer-Verlag.
82. Maedche, A., S. Staab, Stojanovic, R. Studer, and Y. Sure. *SEAL - A framework for developing SEmantic portALs*. in *Proceedings of the 18th British National Conference on Databases*. 2001. Oxford, UK: Springer-Verlag.
83. Matheus, C.J., G. Piatetsky-Shapiro, and D. McNeill, *Selecting and reporting what is interesting*, in *Advances in Knowledge Discovery and Data Mining*, U.M. Fayyad, et al., Editors. 1996, AAAI Press/The MIT Press: Menlo Park, California. p. 495-516.
84. McGuinness, D., R. Fikes, J. Rice, and S. Wilder. *An Environment for Merging and Testing Large Ontologies*. in *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning*. 2000. Breckenridge, Colorado: Morgan Kaufmann.
85. McGuinness, D., H. Manning, L.A. Resnick, and T.W. Beattie*, FindUR: Knowledge-enhanced online search*. 1998. Available at: http://www.research.att.com/~dlm/papers/findur-chi98.ps.
86. Mena, E., A. Illarramendi, V. Kashyap, and A. Sheth. *OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies*. in *Conference on Cooperative Information Systems*. 2000. Eliat, Israel.
87. Montes-y-Gomez, M., A. Gelbukh, and A. Lopez-Lopez. *Comparison of Conceptual Graphs*. in *Proceedings of the First Mexican International Conference on Artificial Intelligence*. 2000. Acapulco, Mexico: Springer-Verlag.
88. Musen, M.A., R.W. Fergerson, W.E. Grosso, N.F. Noy, M. Crubezy, and J.H. Gennari. *Component-Based Support for Building Knowledge-Acquisition Systems*. in *Conference on Intelligent Information Processing of the International Federation for Information Processing World Computer Congress*. 2000. Beijing.
89. Neches, R., S. Abhinkar, F. Hu, R. Eleish, I.-Y. Ko, K.-T. Yao, Q. Zhu, and P. Will, "Collaborative information space analysis tools," in *D-Lib Magazine*, October 1998, 1998.
90. Oostendorp, K.A., W.F. Punch, and R.W. Wiggins. *A Tool for individualizing the web*. in *Proceedings of the 2nd International Conference on the World-Wide Web*. 1994. Chicago, IL.
91. Ostertag, E., J. Hendler, P.-D. Ruben, and C. Braun, *Computing similarity in a reuse library system: an AI-based approach.* ACM Transactions on Software Engineering and Methodology, 1992. **1**(3): p. 205-228.

92. Peters, R. *Exploring the design space for personal information management tools*. in *Proceedings of the 18th Conference on Human Factors in Computing Systems*. 2001. Seattle, Washington.
93. Pirolli, P. and S.K. Card, *Information foraging*. Psychological Review, 1999. **106**(4): p. 643-675.
94. Pirolli, P. and S.K. Card. *Information foraging models of browsers for very large document spaces*. in *Proceedings of AVI'98, the Working Conference on Advanced Visual Interfaces*. 1998. L'Aquila, Italy: ACM Press.
95. Pirolli, P., J. Pitkow, and R. Rao. *Silk from a sow's ear: extracting usable structures from the Web*. in *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*. 1996. Vancouver, British Columbia.
96. Popsecul, A., L.H. Ungar, D.M. Pennock, and S. Lawrence. *Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments*. in *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*. 2001. San Francisco: Morgan Kaufmann.
97. Powers, S., *Practical RDF*. 2003, Sebastopol, California: O'Reilly & Associates, Inc.
98. Rada, R., H. Mili, E. Bicknell, and M. Blettner, *Development and application of a metric on semantic nets.* IEEE Transactions on Systems, Man, and Cybernetics, 1989. **19**(1): p. 17-30.
99. Resnick, P., N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. *GroupLens: an open architecture for collaborative filtering of netnews*. in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. 1994. Chapel Hill, North Carolina.
100. Rocha, L.M. and C. Joslyn. *Simulations of embodied evolving semiosis: emergent semantics in artificial environments*. in *Proceedings of the 1998 Conference on VirtualWorlds and Simulation*. 1998: The Society for Computer Simulation International.
101. Russell, S.J. and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. 2003: Prentice Hall.
102. Sahuguet, A. and F. Azavant, *Building intelligent Web applications using lightweight wrappers.* Data Knowledge Engineering, 2001. **36**(3): p. 283-316.
103. Santini, S., A. Gupta, and R. Jain, *Emergent semantics through interaction in image databases.* IEEE Transactions on Knowledge and Data Engineering, 2001. **13**(3): p. 337-351.
104. Santini, S. and R. Jain, *Similarity measures.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999. **21**(9): p. 871-883.
105. Schohn, G. and D. Cohn. *Less is more: active learning with support vector machines*. in *Proceedings of the 17th International Conference on Machine Learning*. 2000. San Francisco, California: Morgan Kaufmann.
106. Shahabi, C. and Y.-S. Chen. *Web information personalization: challenges and approaches*. in *Proceedings of the Third International Workshop on Databases in Networked Information Systems*. 2003. Aizu-Wakamatsu, Japan.
107. Shardanand, U. and P. Maes. *Social information filtering: algorithms for automating "word of mouth"*. in *Proceedings of the CHI-95 Conference*. 1995. Denver, Colorado: ACM Press.

108. Sheth, A. *Data Semantics: what, where and how?* in *Database applications semantics: proceedings of the 6th IFIP Working Conference on Database Applications Semantics*. 1995. Atlanta, Georgia: Chapman and Hall.
109. Shirky, C.*, The Semantic Web, syllogism, and worldview*. shirky.com: Brooklyn, New York, 2003. Available at: http://www.shirky.com/writings/semantic_syllogism.html.
110. Shneiderman, B., *Creativity support tools.* Communications of the ACM, 2002. **45**(10): p. 116-120.
111. Shneiderman, B., *Dynamic queries for visual information seeking.* IEEE Software, 1994. **11**(6): p. 70-77.
112. Shneiderman, B.*, Information visualization: dynamic queries, starfield displays, and LifeLines*. University of Maryland: College Park, Maryland, 2000. Available at: http://www.cs.umd.edu/hcil/members/bshneiderman/ivwp.html.
113. Shortliffe, E.H., L.E. Perreault, G. Wiederhold, and L.M. Fagan, eds. *Medical Informatics: Computer Applications in Health Care and Biomedicine*. 2nd ed. Springer Series in Health Informatics, ed. K.J. Hannah and M.J. Ball. 2001, Springer-Verlag: New York.
114. Sintek, M. and S. Decker. *TRIPLE - A query, inference, and transformation language for the semantic web*. in *Proceedings of the First International Semantic Web Conference*. 2002. Sardinia, Italy.
115. Srikant, R. and R. Agrawal, *Mining generalized association rules.* Future Generation Computer Systems, 1997. **13**(2): p. 161-180.
116. Staab, S., J. Angele, S. Decker, M. Erdmann, A. Hotho, A. Maedche, H.-P. Schnurr, R. Studer, and Y. Sure, *Semantic community Web portals.* Computer Networks, 2000. **33**(1-6): p. 473-491.
117. Staab, S. and A. Maedche, "Knowledge Portals - Ontologies at work," in *AI Magazine*, Summer, 2001.
118. Staab, S., A. Maedche, and S. Handschuh. *An annotation framework for the semantic web*. in *Proceedings of the First Workshop on Multimedia Annotation*. 2001. Tokyo, Japan.
119. Staab, S., S. Santini, F. Nack, L. Steels, and A. Maedche, "Emergent Semantics," in *IEEE Intelligent Systems, Trends and Controversies*, Jan/Feb 2002, 2002.
120. Staab, S. and H.-P. Schnurr, *Smart task support through proactive access to organizational memory.* Knowledge Based Systems, 2000. **13**(5): p. 251-260.
121. Stoffel, K., M. Taylor, and J. Hendler. *Efficient Management of Very Large Ontologies*. in *Proceedings of the American Association for Artificial Intelligence Conference*. 1997. Providence, Rhode Island: AAAI/MIT Press.
122. Strehl, A., J. Ghosh, and R. Mooney. *Impact of similarity measures on web-page clustering*. in *Proceedings of the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search*. 2000. Austin, Texas.
123. Takkinen, J. and N. Shahmehri, *Are you busy, cool, or just curious? -- CAFE: A model with three different states of mind for a user to manage information in electronic mail.* Human IT, 1998. **2**(1).
124. Tate, A., *Roots of SPAR - Shared Planning and Activity Representation.* Knowledge Engineering Review, 1998. **13**(1): p. 121-128.

125. Thacker, S., A. Sheth, and S. Patel, *Complex Relationships for the Semantic Web*, in *Spinning the Semantic Web*, D. Fensel, et al., Editors. 2003, The MIT Press: Cambridge, Massachusetts.
126. Thuraisingham, B., *XML Databases and the Semantic Web*. 2002: CRC Press. 336.
127. Tong, S. and D. Koller. *Support vector machine active learning with applications to text classification*. in *Proceedings of the 17th International Conference on Machine Learning*. 2000. San Francisco, California: Morgan Kaufmann.
128. Tversky, A., *Features of similarity.* Psychological Review, 1977. **84**(4): p. 327-352.
129. Vdovjak, R. and G.-J. Houben. *RDF-Based Architecture for Semantic Integration of Heterogeneous Information Sources*. in *Workshop on Information Integration on the Web*. 2001.
130. Vega, J., A. Gomez-Perez, A. Tello, and H. Pinto. *How to find suitable ontologies using an ontology-based WWW broker*. in *Proceedings of the International Work-Conference on Artificial and Natural Neural Networks*. 1999. Alicante, Spain: Springer.
131. Ware, M., E. Frank, G. Holmes, M. Hall, and I.H. Witten, *Interactive machine learning - letting users build classifiers.* International Journal of Human Computer Studies, 2000. **55**(3): p. 281-292.
132. Watson, I., *Applying Case-based Reasoning: Techniques for Enterprise Systems*. 1997, San Francisco: Morgan Kaufmann Publishers, Inc.
133. Wexelblat, A. and P. Maes*, Footprints: Visualizing histories for web browsing*. MIT: 1997. Available at: http://web.media.mit.edu/~wex/Footprints/footprints1.html.
134. Wiederhold, G. and M. Genesereth, *The conceptual basis for mediation services.* IEEE Expert / Intelligent Systems, 1997. **12**(5): p. 38-47.
135. Williams, M.D., *What makes RABBIT run?* International Journal of Human-Computer Studies, 1984. **21**(4): p. 333-352.
136. Williamson, C. and B. Shneiderman. *The dynamic HomeFinder: evaluating dynamic queries in a real-estate information exploration system*. in *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*. 1992. Copenhagen, Denmark.
137. Winston, P.H., *Artificial Intelligence*. 1992: Addison-Wesley.
138. Witten, I.H. and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. 2000, San Francisco, California: Morgan Kaufmann Publishers.
139. Yao, K.-T., R. Neches, I.-Y. Ko, R. Eleish, and S. Abhinkar. *Synchronous and asynchronous collaborative information space analysis tools*. in *Proceedings of the International Workshop on Collaboration and Mobile Computing*. 1999. Fukushima, Japan.
140. Youssif, R.S. and C.N. Purdy*, Fuzzy similarity measures for signal pattern classification*. ECECS Department, University of Cincinnati: Cincinnati, Ohio, 2003. Available at: http://www.ececs.uc.edu/~fit/MAICS/PAPERS/Roshdy.pdf.