Tools Designed Specifically for Information Space Organization

Stephen Davies
Roger King

Department of Computer Science
University of Colorado at Boulder

Technical Report CU-CS-998-05

August 2005

Tools Designed Specifically for Information Space Organization

Stephen Davies, University of Colorado[1]
Roger King, University of Colorado

Abstract

Information retrieval has traditionally been concerned with extracting isolated facts from large knowledge bases. There are times, however, when users have a different goal – they seek to understand the structure of the data as a whole, so that they have a conceptual framework into which individual facts can later be assimilated. Today's search tools do not explicitly support this. They typically provide only a query-response mechanism for obtaining isolated facts, one at a time. Users are forced to form a mental model on their own, inferring the broader structure from these disconnected details. Tools specifically designed to help users organize large information spaces are proposed. Such tools will raise the level of abstraction, so that the categories of mental perception can be stated, examined, and reasoned about in their own right. The user's subjectivity will be honored; different people often have different views of the same data, and they want to organize it according to their own individual perceptions. The need for such tools, and the properties designers should strive for, are discussed. A prototype that demonstrates the validity of this kind of tool is presented.

I. Introduction

Traditionally, interfaces to large databases have been steeped in the query-response model. Whether formulating SQL queries or performing Google searches, the user's procedure is the same: issue a request for some specific piece of information, and receive a specific result. The hope is that with enough iterative refinements, the user can successfully extract the particular bit of knowledge they were looking for. Tremendous effort has gone into improving the mechanism by which users can ask such questions, and finding ways to deliver more relevant results.

But consider the query-response paradigm for a moment. 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 task (and 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.

[1] Contact author: Stephen.Davies@colorado.edu.

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

But there are times when we users have a completely different goal in mind, and we end up using a search engine like Google to accomplish it indirectly. Sometimes we want to really *understand* some part of the information space in its entirety – what kind of information is out there on a given topic, what may be missing, and how it is all related. And so we form a mental model that helps us generalize and give structure to the myriad of individual facts. This allows us to draw broader conclusions about what the data is telling us as a whole, and gives us a framework in which to assimilate new facts as they arrive.

Perhaps we're interested in buying a new disk drive, but we 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 we 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 we're seeking much more than definitive answers to a few isolated questions. Instead, we're building up a conceptual model as we go – gathering information, relating it to what else we know, making judgments about how it is interrelated, and gradually forming a complete mental picture of how the information is assembled as a whole.

Google, of course, has no idea that we're trying to do this. It sees no relationship between our various queries, and has no concept of what we're really trying to find out. Each query is a standalone event that provides some piece of information detached from everything else, and it is up to us to assemble these bits ourselves into a meaningful mental framework. The search engine is rather like a consultant who is supremely brilliant and utterly dull at the same time: he knows all the answers to every individual question we might ask, but throughout the interview it never dawns on him what we're really driving at. He coughs up detail after detail, but sees no relationship between them and can help us see none.

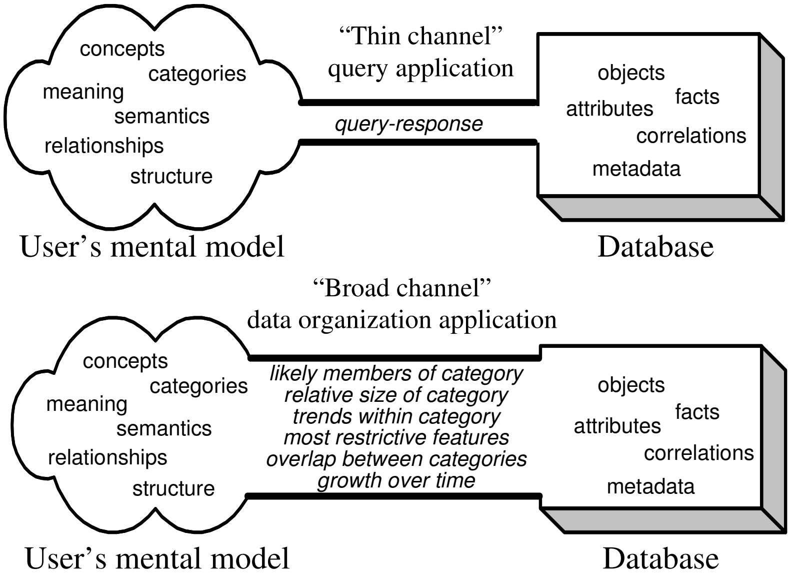

Figure 1. Two different approaches in database interfaces. In the top picture, each database interaction is an isolated question and answer, resulting in a single isolated fact. The user is forced to infer the broader structure of the information space by manually assembling these facts into a mental framework. 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.

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 understanding is what gives experts the background to render individual detailed decisions anyway. A prospective homeowner goes to a realtor in part because he believes the end result will be a particular property that is well-suited to him. But what gives the realtor the ability to recommend such a house is that she is able to draw on a deep understanding of what kinds of properties are available in the area, which ones might be best suited to her client, what issues are involved that he may not have considered, and so on. The final recommendation 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 she can provide many such recommendations. And it is hardly ideal to try and build such a comprehensive understanding by selecting straws from a haystack one at a time.

In short, the typical information retrieval application does not offer us any explicit way to track or organize or predict or evaluate the emerging conceptual structure that we build 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.

II. Categories for Organization

We propose to design tools 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 herself, providing her with only a sequence of facts, we aim to incorporate her mental constructs explicitly into the user interface experience. We 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 her in accomplishing her *real* goal.

In concrete terms, a first step towards implementing this is to add the notion of *categories* to the user's browsing toolkit. As people traverse a large information space (such as the Web, or a large digital library) they form semantic judgments and naturally group together pages they view as similar. Perhaps without realizing it, they are categorizing or clustering what they encounter, in order to reduce complexity and impose their own rational order on what is otherwise an overwhelming amount of information [Peters 2001].

One way to represent "the user's mental model," then, is simply as the set of categories that the user has identified. Each category consists of a number of instances (e.g., Web pages) that the user has explicitly associated with it. (These categories may be overlapping, since a given instance may belong to multiple categories.) A category is something much more than just a group of specific instances, however. It is an abstract, mental concept that the user has in mind, of which the labeled instances are merely examples. The concept endures outside the context of the specific instances. It is a semantic entity, representing some real-world phenomenon that the user believes is important enough to use in imposing structure on the entire information space.

Electronic tools, of course, have no access to the real-world semantic concept. They can only see the instances that the user has denoted as representative. But if those instances can be represented by some sort of structured feature set, the system ought to be able to use standard data mining techniques to reason about the underlying definition of the category.

This is the heart of our information space organization approach. Humans excel at rendering intuitive judgments about real-world entities. Data mining excels at examining large numbers of instances that have been so judged, and discovering statistical generalizations about them. By designing an exploration tool that enables both of these to take place, we provide powerful leverage to systematize a large set of data. Users can browse a small part of the space, recognizing categories and grouping related instances together. The system, then, can introspect and guess at a "definition" for each category in terms of the underlying features of its instances. This category definition, perhaps after some editing, can then be used to organize and answer questions about the larger information space that is far too time-consuming for the user to explore in detail manually.

A. An example scenario

Imagine a professional caterer who has his entire recipe base available in an online database system, perhaps an intranet of Web pages that can be browsed. This information space is absolutely vital to his business, since it allows him to appropriately plan events, satisfy his clientele, and minimize costs. Different types of meals are appropriate for different venues, and have different cost profiles based on the number served. This caterer often serves conventions or other multi-day events, and requires sequences of meals that offer variety and balance of tastes, while at the same time meeting competitive budgets.

Each web page contains a description of a single recipe, including its ingredients, preparation time, portion sizes, cost, equipment required for preparation and serving, and so on. Whether this information is expressed in free text or in a structured format, it can be mined to produce a set of features that represent that recipe. These features are the material that data mining algorithms will operate on to assist the caterer in organizing his information space.

This professional brings considerable intuitive insight to his task. He may have hundreds or even thousands of possible entrees to select from, but of course his mental conception of them is not merely an unorganized miscellany. He knows that there are certain *types* of recipes appropriate to certain circumstances, from which he can mix and match. For an elegant winter wedding reception, he immediately thinks of *Beef Wellington* or *Steak au Poivre*; but with an all-day conference involving three full meals on a constrained budget, he would suggest *Prosciutto Sandwiches* or *Pineapple Chicken Wraps* for a light outdoor lunch offering.

An information space organization tool allows him to express his intuition about these categories more formally, and brings an inductive reasoning system to bear on the problem. For example, he could group several recipes together in a category called "light lunch fare," and then ask the system for a preliminary definition of that category. Data mining algorithms would examine the features of the selected instances, contrasting them with the values in the recipe pool as a whole. The system would then display the statistically significant commonalities. In effect, the system is reporting, "based on the fact that you included these seven examples in your category, here's my best guess as to what you mean by 'light lunch fare.'"

The caterer may discover, for example, that the recipes he has classified under "light lunch fare" tend to have short preparation times, mayonnaise as an ingredient, and under four hundred calories per serving. This information may be helpful in its own right, motivating the caterer to recognize and reflect on his own tendencies and biases. But it also allows him to raise higher-level questions about the category as a whole, as follows.

He would first edit the system's preliminary definition, refining its criteria and removing obviously spurious attributes. Then, once he is satisfied, he could ask the system questions like:

- "What are some 'light lunch fare' recipes that are vegetarian?" This returns a list of instances that fit these criteria.
- "What percentage of my entire recipe base could be classified as 'light lunch fare?'" The system may respond that 17% of his entrees match this general pattern.
- "How do 'light lunch fare' recipes compare with others in cost?" He may learn that recipes that fit this category tend to be nearly a dollar less per serving than the average recipe.
- "Which 'light lunch fare' recipes haven't been served in the last year?" Assuming archival information is available, he will obtain a list of alternate dishes that he may have forgotten about, but which might be worth rotating back in to his repertoire.

Eventually, as the caterer continues to use the tool to explore his information space, he will forge a set of important categories that help him get a handle on it. He may create categories of recipes for “multi-course formal dinners,” “sit-down lunch parties,” and “large outdoor gatherings,” populating each one with representative instances. He can then view his entire library at a higher level of abstraction, reasoning in terms of categories rather than individual instances. The mental model that he needs to survive is no longer manually cobbled together as a result of a sequence of queries, but is rather tracked by the tool itself, which assists him in creating, defining, and extrapolating from categories. And since this intuition and expertise is no longer isolated in his mind, it can be shared with other users who can learn from and perhaps modify these categories to suit their own purposes.

B. Properties of effective tools

We have identified a number of desirable properties for information space organization tools. Specifically, we believe that they should be abstract, subjective, proactive, persistent, and augmentative.

- **Abstract**. The goal is to identify patterns and global structure in the data; hence, tools must allow the user to raise the level of abstraction above that of individual instances. Humans should be able to use the intangible concepts in their minds as first-class constructs in the interface. This means that in addition to reasoning about concrete notions such as “entrees with preparation time of under twenty minutes,” they must also be able to reason about abstract notions like “entrees appropriate for intimate dinner gatherings.”
- **Subjective**. In any non-trivial information space, there are alternate ways of organizing data, each based on an individual’s perception, and each of them equally “correct.” When visualizing items in a grocery store, some may associate the cereal with the granola bars (based on food content), and others may group it with the coffee (based on meal type.) Neither of these organizations is more valid than the other. They both identify significant patterns in the data that may be of value. Indeed, some might want to make use of *both* groupings, depending on the context and the task at hand. We contend that the user must be free to organize the information space in any way she sees fit, based on the intuitive conceptions, priorities, goals, and biases she brings to the problem.

  This is very different than traditional clustering methods. Many attempts have been made to automatically categorize a large group of documents or data records, based solely on their objective features. These approaches do add value by bootstrapping some semblance of structure into an otherwise impenetrable morass. But they give little opportunity for an information seeker to express her own understanding of how the data should be organized. Hence any clusters that are revealed are hardwired to whatever semantics the authors of the data had the foresight to add.

  We would argue that there is really no such thing as “the” semantics of a set of data. Data is just data. Semantics only emerge as a user introduces her own subjective evaluation to it. The system’s job, then, 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 her.
- **Proactive**. The user is ultimately in charge of the browsing experience, but an information space organization tool can and should suggest what to look at next. The system can quickly scan large quantities of data in search of items pertinent to the current category. This helps shape the browsing path towards objects of interest, relieving the user of the burden of having to know where to look for relevant information. The system can also detect large groups of instances that are quite unlike any category the user has created thus far. It can then direct the user’s attention to this unfamiliar territory at the appropriate time, so that she is aware of its existence and can begin to see how it contributes to the total information space structure.
- **Persistent**. The information gained from a user’s interaction with the system should be captured and stored. When the user returns, then, it will not be up to her to “dust off the cobwebs” and try to mentally reconstruct what she 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, she endows it with semantics. In some cases, these semantics may be broadly applicable beyond the particular user who identified them, and it may be worthwhile to annotate them explicitly in the data. The system should assist the user in determining when a perceived relationship is worth being denoted in the database proper, and provide an easy mechanism for doing so. As more users explore the system, more semantic annotations are added, and the better the system is able to identify real-world relationships.

III. "Horse": A prototype tool for information space organization

To demonstrate these ideas, we developed a prototype system whose core concepts will form the foundation of our future work. The system, called "Horse," features only a command-line interface since our present goal is simply to validate our basic approach. But it implements a key part of the framework outlined above, proactively guiding the user through an information space organization experience.

This section has two parts. First, we will describe the prototype in general terms: the nature of the data it operates on, the flow of the user interaction, and the preliminary machine learning algorithms it employs. It is important to realize that all of these aspects are extremely speculative at this stage. The purpose of our prototype is not to prove the superiority of any particular technique, but simply to demonstrate one core aspect of a possible information space organization paradigm. Second, we present our empirical findings from running the system on a nontrivial data set: a digital library of popular films.

A. The prototype

1. The data

Horse can operate on any data for which low-level features have been extracted in the form of key-value pairs. The information space must be comprised of individual instances, each of which is described by a set of keys and their associated values, with possibly multiple values corresponding to a given key. A feature is simply a key and its value taken together. To continue the culinary example, the recipe *Tuna Salad Sandwich* would include the features "ingredient=mayonnaise" and "ingredient= tuna," while *Chicken Noodle Soup* would not.

At the heart of the system is the ability to reason about the semantic similarity between various instances. Given the fact that the user has grouped several instances together into a category, the system must be able to form a judgment about the likelihood that any *other* particular instance might also be a member of that category. To do this, the system makes use of the only information it has: the features of the instances. In general, the more features an instance has in common with those of a category's members, the more we consider the instance to be a likely member of that category.

2. The interface

The goal of the system is to help the user transition from chaos to order. The input is a disarray of instances; the end result is a set of user-defined categories that organize and give structure to those instances.

Since the purpose of the system is to create and refine categories, this is the backdrop against which all user interaction takes place. The user has at his disposal a set of named categories that he has created, and can switch between them at any time. Only one is active at any given time; it is called the "active category." This is the category that the user is currently working with, and which the system uses to select instances that may be of interest. Seeing a particular instance, however, may cause the user to want to "switch gears" to a different category, or create a new one altogether.

The basic interaction paradigm involves viewing instances and providing feedback. The system selects an instance from the information space (as described below) and displays it to the user. The user renders a judgment about it: specifically, whether it is or is not a member of the active category. He expresses this to the system by choosing either the "in" or "out" command.[2]

For each category, the system maintains two sets of instances: the known examples and the known counterexamples. Every time the user indicates that an instance is a member of the category he has in mind, Horse includes it in the set of known examples for that category. Every time he indicates an instance is *not* a member, Horse includes it in the set of known *counter*examples. Thus as the user browses, the system gathers more and more information about what kinds of instances satisfy the user's semantic criteria, and what kinds do not. After each judgment, the system selects a new instance to display, and the process continues. Each time the user views an instance and then makes a judgment is called an *iteration*.

3. The instance selection algorithm

The choice of which instance to show the user next is not merely random. It is carefully selected by the system based on what is known about the active category so far. The current algorithm works as follows: first, from the instances in the information space that the user has not yet seen, randomly select a small subset (say, twenty.) This is called the *quorum*. Then, examine each instance in the quorum and evaluate how likely it is to belong to the active category. Finally, show the user the instance that is most likely to belong.

The astute reader will recognize that determining the likelihood that an instance belongs to a category is very similar to a standard machine learning problem. At any point in time, we have two sets of instances that the user has provided, one labeled "in the category" and one labeled "not in the category." This is our training data. We predict category membership for a new instance by simply training a classifier on the labeled instances and using it to classify the new instance as either "in" or "out." We currently use the familiar Naïve Bayes algorithm to achieve this since it is fast and simple.

Typically, a two-label Naïve Bayes classifier is used to first estimate the probability that an instance should be assigned label 1, and then the probability that it should be assigned label 2. These two probabilities are compared, and whichever one is greater determines the classification. Our situation is somewhat different, since our goal is not to actually label an individual instance, but rather to predict *which* of a quorum of instances is most likely to have a particular label (namely, the "in the category" label.) We accomplish this in a straightforward way: for each instance, we subtract the probability that it is a counterexample from the probability that it is an example, and arrive at a single score. Whichever instance in the quorum has the highest score is the one we choose to display.

We make several further simplifying assumptions in implementing Naïve Bayes. Given an instance with features $\{F_i\}$, the standard formula assigns the probability that it should be assigned label X as:

$$P(X \mid F_1, F_2, ..., F_k) = \frac{\prod_i P(F_i \mid X) \times P(X)}{P(F_1, F_2, ..., F_k)}$$

Since we are seeking the *relative* likelihoods for each of two labels, rather than strict mathematical probabilities, we drop the denominator in this calculation because it is constant. But for a similar reason we also drop the "prior" term P(X). This is normally estimated from the overall proportions of the labels in a

[2] Other actions are possible during an iteration besides simply giving "in" or "out" feedback on the current instance. As mentioned above, the user may choose to switch categories based on seeing the new example. ("No, I wouldn't consider this item 'light lunch fare,' but it makes me realize that I probably need a different category for 'outdoor barbeque items,' so let me start to work on that…") The user can also request to view and possibly edit the current category definition, as explained below.

representative tagged corpus. It represents the likelihood that any random instance would be a member of the category. [3]

There are two reasons we omit this factor. First, we do not have any reliable information about it. The only instances we know anything about are the ones the user has explicitly labeled, and since we have deliberately been steering him towards presumed "in" examples, these are hardly representative. But beyond this, we actually don't even care about P(X): we are only comparing examples within a quorum, not attempting to label them outright. Regardless of how strict a category's criteria are overall, we would draw the same conclusion about which of a quorum of instances was most likely to belong to it. Hence, the "score" that we compute for an instance with features $\{F_i\}$ reduces to:

$$Score(F_1, F_2, ..., F_k) = \prod_i P(F_i \mid "in") - \prod_i P(F_i \mid "out")$$

The individual factors in these two products are computed in the usual way: as a fraction of the number of labeled training instances that have the given feature.[4] We also add a constant smoothing factor to eliminate zero counts. The overall instance selection algorithm is captured in Figure 2.

This technique is experimental, and we plan to adjust it in light of what we learn from our investigations. But it is worth noting that the "quorum" approach attempts to mimic what humans ordinarily go through as they form categorizations about things in their world. People excel at forming useful generalizations from only small subsets of the available data. Consider the film domain. Anyone who has seen a fair number of movies during their lifetime inevitably groups them in various ways – they know that there are "violent action movies" and "Hitchcockian thrillers" and "silly comedies." These are doubtless real-world phenomena, yet the amazing thing is that we can identify them by viewing only a small fraction of the total number of movies ever produced. This is the rationale behind our "quorum" approach. The system achieves scalability in the same way humans do: by being satisfied to examine very small samples at a time. To determine what a human means by a particular classification, Horse grabs a few examples that are "close at hand" and presents the one in that group that looks the closest to what they have already identified. It doesn't have to search the database exhaustively at each iteration, looking for the very best match out of millions of items. After all, one doesn't need to watch every movie ever filmed to determine that there is such a thing as "silly comedies."

Notice the role of the N_q parameter in this procedure. As it shrinks to 1, the system presents instances more and more completely randomly. As it grows to *N*, the size of the entire information space, the more the system scours the database looking for what it thinks is the closest possible fit to the user's category. In between these two extremes, the system presents an instance that looks "good enough": better than merely a random selection, but also likely to be inferior to a number of other instances that would have been found had the database been exhaustively searched.

This represents a compromise between two conflicting goals. On the one hand, we want to honor the user's time by showing him what we feel is most relevant to him. On the other hand, we also want to learn as much about his category definition as we can, and this is best achieved when we get corrective feedback in the form of key counterexamples. If we showed him only "safe" examples that we were certain he would approve for the category, the system's learning would be slowed. Showing the user the best of a randomly selected pool, then, balances these two objectives.

[3] For example, if nearly every recipe in the database could be considered "light lunch fare," then the prior probability P("in") would be very high and P("out") would be low. If such recipes were relatively rare, on the other hand, the opposite would be true.

[4] For instance, if five out of the eight instances identified as "light lunch fare" had "ingredient=mayonnaise" as a feature, then P("ingredient=mayonnaise" | "in") would be 0.625. If only one of the ten instances that the user has determined are *not* "light lunch fare" had that feature, then P("ingredient=mayonnaise" | "out") would be 0.1.

At the start of each iteration, we have an active category that has a set of known examples {E$_i$} and a set of known counterexamples {C_i }.

1. Randomly choose a quorum of N_q instances from the entire information space, {Q$_i$ }, 0 <= $i < N_q$.

2. For each instance in the quorum, use its features to compute the likelihood score that the user would judge it as an example of the active category:

$$Score(F_1, F_2, ..., F_k) = \prod_i P(F_i | "in") - \prod_i P(F_i | "out")$$

(Each factor in the two terms includes an add-constant smoothing factor to eliminate zero counts.)

3. Show the user the instance with the highest score, and ask for feedback as to whether this instance is in fact an example or a counterexample.

4. Add the new instance to the example set or counterexample set, appropriately, and repeat these steps for the next iteration.

Figure 2. An instance selection algorithm.

4. Hypothesizing about category definitions

Finally, as the user adds examples and counterexamples to a category, Horse attempts to "learn" the underlying definition. This amounts to a feature selection process in which the system determines which features are truly relevant to distinguish members of the category from nonmembers. For this, we use a statistical difference of means test to trim features that do not meet a certain significance threshold.

At each point in time the active category has N_e examples {E_i} and N_e counterexamples {C_i}. Horse considers each feature in turn. Every feature is either present or absent from a given instance. We compute the "mean" of each of these two groups (the examples and the counterexamples) by assigning a value of 1 to an instance if it has the feature, and 0 if it doesn't. Call these means μ_e and μ_c. Our significance statistic is:

$$Significance = \left| \frac{(\mu_e - \mu_c)}{\hat{\sigma}_{M_D}} \right|$$

where $\hat{\sigma}_{M_D}$ is the estimated standard error of the difference between means, or

$$\hat{\sigma}_{M_D} = \sqrt{\frac{\hat{\sigma}_e^{\,2}}{N_e} + \frac{\hat{\sigma}_c^{\,2}}{N_c}}$$

($\hat{\sigma}_e^2$ and $\hat{\sigma}_c^2$ are the estimated variance of the examples and counterexamples, respectively, with regard to the feature.) We establish a threshold for this statistic, and trim from consideration any feature which fails to meet it. Since the data is sparse, we use the "student's t" distribution to vary the threshold based on the sizes of the groups. (As more examples and counterexamples are acquired, the raw value of the statistic does not have to be as high for the feature to be considered significant.) Features which meet the threshold, then, we deem to be relevant to the category. They are retained in what we call the "hypothesis"; that is, the system's current best guess as to what the true underlying category definition is.

For practical reasons, we do not attempt to trim features until the sizes of the category's example and counterexample sets reach a certain critical mass. Currently we have this set at four: when at least four examples and four counterexamples have been identified, we apply the difference of means test and eliminate features that do not seem to affect category membership. From that point on, we re-apply the test at the end of each iteration so as to gradually prune irrelevant features from our hypothesis.

At any time, the user can request to view the current category hypothesis. This displays a list of all of the features the system deems relevant, together with their significance values. The user can review and edit this feature list, directly adding or removing features based on his intuitive domain knowledge.

B. Organizing the Internet Movie Database

The Internet Movie Database (IMDb, at www.imdb.com) is a free, online collection of movie information whose maintainers encourage public contributions of data. It is massive, containing structured and free-text information for over six million different films. Users can browse selections online and get information about all of a movie's screen credits (directors, actors, producers, etc), as well as more subjective information such as aggregate user ratings and descriptive keywords.

IMDb also permits its data to be downloaded as a series of text files. To test our prototype, we obtained a small subset of the total database and massaged it into the format that Horse can understand. We chose to concentrate on the "top 250 movies" as judged by overall IMDb user ratings, and on only a small set of features: the writers, directors, and descriptive keywords for these films. Altogether, this amounts to around one megabyte of text information. Horse reads this data from disk at startup and then allows the user to explore it as described above.

The information is almost ideal for our study. Since we include only the highest rated films of all time, users are likely to have at least heard of nearly every instance they are presented with. Yet it is diverse enough that different users are likely to want to categorize the films in different ways, depending on their perceptions and tastes.

We attempted to include only features that would have a bearing on a movie fan's perception of similarity. It seemed unlikely that a user would consider two films similar because they had the same boom operator, for example. But two films with the same writer or director might well be perceived as similar for that reason. The keywords are a particularly interesting case. Each IMDb film contains a variable number of annotations that online users have added in an attempt to describe some feature of the plot. For example, the keywords for the 1957 film "Witness for the Prosecution" are:

based-on-play courtroom twist-in-the-end blood-type microphone pill trial will-and-testament perjury murder remake based-on-adaptation

This is a relatively small list; "The Godfather," for instance, contains well over a hundred terms. Our supposition is that when a user groups several films together in a category, the similarity that he perceives may be reflected in some subset of shared keywords. If true, our system should be able to detect the relevant ones and use them to characterize the category as a whole.

1. A sample scenario

We now present the results of an actual interactive session with Horse, in order to illustrate how the system can be used.

A father of young children has recently rented "The Wizard of Oz" from his local video store and shown it to the entire family. The children were delighted with the film, and so were the parents: it was entertaining, provoked family discussion, and featured content that they felt was appropriate for the age of the viewers. A regular family movie night, they decide, might be an enjoyable activity to incorporate into their weekly routine, provided that they can find other movies as suitable as "The Wizard of Oz" turned out to be.

The father returns the video to the store, and on the way, he considers what he might rent for upcoming movie nights. Perhaps without realizing it, he is beginning to form an abstract category in his mind: "Films for Family Movie Night." Of all the movies ever produced, only a relatively small subset would meet his criteria. Yet it is difficult for him to even state what the criteria are – they are not directly related to any single attribute that he knows of; for instance, a particular director, actor, or film studio. The only thing the man has to go on is his knowledge of particular instances. He remembers seeing "Oliver!" and "Cinderella" as a child, and knows that they would be appropriate, whereas "Saving Private Ryan" and "Silence of the Lambs" would obviously not be. But as reliable as this intuition may be, it is just that: intuition. The father has no idea what features any of these films have. Although he knows that "Oliver!" fits the category he has in mind, he could not say who the director or any of the actors were, or what descriptive keywords anyone might have assigned to the film. All he can do is form semantic judgments about each instance as a unit. It is up to Horse to help him discover which features may bear out his intuition.

He engages the application and creates a new category called "Films for Family Movie Night." Then he bootstraps this category by finding the particular instance "The Wizard of Oz" and adding it as an example. (Finding a known film by name is achieved by a straightforward Boolean search.) Horse then begins the first iteration for this category. It collects a quorum of twenty random samples from the entire pool, judges each of their probabilities of belonging to the category (based only on this single example, of course), and displays for the user the film with the best score:

Braveheart.

At first glance, this seems preposterous. What movie could be more different from "The Wizard of Oz" than a violent war epic like "Braveheart?" To understand this, we must recognize three factors. First, remember that "Braveheart" merely represents the most likely instance in the *quorum*, not the entire database. It is always possible to get a "bad sample" with very few representative instances, in which case the best of a bad lot will be shown. This is actually beneficial, since it helps the system learn by providing an opportunity for a counterexample. Second, upon examination of the low-level features, it turns out that the two films actually do share a number of keywords, among them: *uncle*, *castle*, *courage*, and *friendship*. Presumably some or all of these will turn out to be statistically insignificant as more data is acquired about the category. And this is precisely the third factor: the information about the category is yet small. When the user first begins, and has specified only one or two examples or counterexamples, the system has very little to work with. It weighs spurious attributes as heavily as truly relevant ones, since it cannot tell the difference. Hence, it is quite likely to veer off in the wrong direction at first. As more and more examples and counterexamples are identified, however, we expect Horse to converge on the "true" category definition.

And this is in fact what eventually happens. After the user resoundingly informs Horse that "Braveheart" is *not* an example of the "Films for Family Movie Night" category, the system adds the instance to its set of counterexamples and performs a new iteration. The next few instances presented, together with the user's judgments are, in order:

The Terminator (out)
Pirates of the Caribbean (out)
Beauty and the Beast (in)
Young Frankenstein (out)
A Beautiful Mind (out)
Back to the Future (in)
Taxi Driver (out)
Hobgoblins (out)
X2 (out)
Future War (out)
E.T. The Extra Terrestrial (in)
From Justin to Kelly (out)
Stalag 17 (out)

Each entry in this list represents an iteration of the system: Horse chose to display the instance listed, and the user responded with feedback as to whether the instance was "in" or "out" of the "Films for Family Movie Night" category.

So far, mostly counterexamples have been presented, which indicates that the category is not yet converging. Several of the counterexamples were close enough to give the user pause, however. "Pirates of the Caribbean," a recent Disney film, seems like a possibility until the man remembers overhearing at work that it contained some scary images, and therefore might not be appropriate for his children. Similarly, "Young Frankenstein" is a fairly close match, except that the man has seen this film before and recalls some adult humor. So he rejects this instance as well.

It may seem that up to this point the system has been floundering, presenting merely random examples and determining nothing useful about them. But in fact our experience with Horse is that once a critical mass of examples and counterexamples have been identified, the system tends to converge quite rapidly. So it is here. In an almost magical way, the next sequence of examples presented are:

It's a Wonderful Life (in)
The Adventures of Robin Hood (in)
North by Northwest (out)
Star Wars: Return of the Jedi (in)
Fantasia (in)
Toy Story (in)
Being There (out)
The Princess Bride (in)
Finding Nemo (in)
Whale Rider (in)
Stand by Me (in)
Toy Story 2 (in)
Monsters, Inc. (in)
Shrek (in)

and so on. At this point, nearly every example Horse chooses to show is exactly what the user has in mind. Clearly, it has caught on to something in the underlying data that reflects the user's mental perception. The user requests to see the current category hypothesis, and is presented with:

Current hypothesis for category "Films for Family Movie Night":
NOT keyword=murder (2.40)
keyword=cgi-film (2.70)
keyword=computer-animation (2.70)
NOT keyword=marriage-proposal (2.40)
keyword=kids-and-family (7.20)

Each feature listed here represents a statistically significant tendency in the data that the user has judged. Those preceded by "NOT" tend to be reflective of counterexamples; the others are reflective of examples. The numbers in parentheses are the values of the significance statistic, as explained above.

Clearly, the most relevant feature is the keyword "kids-and-family"; it was present in 14 of 16 examples, and in none of the 13 counterexamples. This may seem obvious; however, the user did not know *a priori* that such a feature even existed. To discover it manually, he would have had to wade through lists of literally hundreds of keywords and other features hoping to guess which one(s) would be applicable to his category. Instead, Horse automatically sifted this out for him by simply reasoning about the examples he had identified.

The other features are thought-provoking as well. The keyword "marriage-proposal" looks like an accidental artifact of the data, one which the user will probably trim. (For the curious, it is present in the counterexamples

"Braveheart," "Pirates of the Caribbean," "A Beautiful Mind," and "North by Northwest," but in none of the examples.) Computer animation is something he had not previously considered, but this discovery makes him realize that several of the instances he has identified are in fact cartoons, and that searching specifically in that direction for movies might yield a fruitful search. "Cgi-film" is a term the user is not familiar with, but after a bit of poking on the Web he discovers that CGI stands for "Computer Generated Images," which again reinforces the previous discovery.

2. Discussion

This example, though simple, illustrates a number of benefits to this kind of user interaction paradigm:

1. The user is relieved of the responsibility of knowing how the data is represented. In our scenario, the user didn't have to study the features of the instances in order to effectively interact with the database. He simply operated at the instance level, and let Horse worry about which features were relevant to his inquiry. Then, when he asked for the category definition, he was instantly directed to the probable features of interest. This is an especially important benefit when novice users are involved. They approach the database with considerable uncertainty as to what it contains, what they want to know, and how to ask for it. This style of interface enables them to be productive with very little up front time investment.

2. It helps the user clarify and crystallize his intuition. The user chooses instances based on his semantic interpretation, which may be somewhat ill-defined. But the system promptly informs him of the feature-level trends that align with these choices. In the above scenario, Horse exposed what the man perhaps hadn't realized: that in general, CGI films were likely candidates for movie night. The interface experience is thus not only productive, but insightful.

3. It enables customized category-based information retrieval. As we have seen, once the system has enough examples to converge, it can readily find new instances of a category using its similarity metrics. This is potentially much more precise than a hand-crafted Boolean search, in which a user must try to express his intuition as a combination of low-level fields. With Horse, finding new instances "like these" happens automatically.

4. In enables category-based questions to be posed. Our current prototype does not demonstrate this, but it is a major aim of our future work: to raise the level of abstraction from instances to categories. One could imagine the man in the above scenario asking for the common writers and directors of "family movie night" films, or which movies in that category overlap with another category, or how frequently "family movie night" films are released these days compared with in decades past. Indeed, we believe it is often the case that the questions users want to ask about a large database are general and strategic, rather than concerned with individual details. And once a user has refined a set of categories to his satisfaction, they can be used as operands to powerful category-level operators that shed light on the database's trends as a whole.

IV. Related work

The problem of automatically clustering large groups of documents or structured instances has been given much attention. Advanced web exploration tools like KartOO [Baleydier 2004] and [Grokker 2004] perform this for natural language text, for instance, and the On-To-Knowledge project's Spectacle initiative [Fluit et al. 2003] does the same for ontologically annotated data. What makes these approaches different from ours is their implicit objectivity. They analyze data and group it according to the system's absolute definition of similarity. Such tools provide important benefits, since they reveal structure that was previously buried. Our belief, however, is that great gains can be made by incorporating the user's subjective notions as well. Instances may have a multitude of features, but for a given user executing a given task, only a few may be relevant, and in ways that a static clustering algorithm would be unable to predict. Spectacle recognizes this to some degree: 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.

The SEAL portal from the University of Karlsruhe [Maedche et al. 2001] performs a "semantic ranking" of ontology-based query results based on the perceived similarity to the posed query. This incorporates ideas such as the relative proximity of terms within a taxonomy. This bears some resemblance to our judgments about likely category membership, though we envision permeating these ideas throughout the user interaction experience rather than using them in an auxiliary role.

Personalized search systems, such as those described in Shahabi's excellent survey [Shahabi and Chen 2003], take into account the interests of users when filtering query results. The emphasis, however, is on sifting "irrelevant" information from what the user finds "interesting." In our approach, categorization is a different problem from interesting vs. non-interesting – it is the discovery of different *kinds* of interesting things, and how those different kinds relate to one another.

Closer to our approach is the DASHER project from the University of Southern California [Neches et al. 1998; Yao et al. 1999]. Users can collaborate to organize a web information space, creating hierarchical categories to which specific URLs can be added. Natural language processing algorithms help identify common noun phrases in documents, so that users can compare their own categorizations with the noun phrase clusters. Inductive learning then attempts to populate categories with instances based on "bag of words" comparisons. The system builds decision lists to classify documents, and then uses a "fan out" technique to examine the hyperlinks emanating from a page to search for other similar pages. In general, DASHER 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. Our work can be seen as extending these ideas, moving towards an interface in which categories are more than merely groupings of pages, but first-class objects about which higher-level questions can be posed.

Santini et al.'s work on image databases [Santini and Jain 1999; Santini et al. 2001; Staab et al. 2002] also resembles ours to some degree. Their 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 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. This work differs from ours primarily in that it is custom-tailored to the image domain, and also does not support the category-based questioning so central to our approach.

V. Conclusion

The use of semantically sophisticated tools for mining, filtering, and querying vast amounts of information on the Web has been studied in detail [Boley et al. 1999; Chakrabarti and Batterywala 2000; Cooley et al. 1997; Grosky et al. 2002; Lieberman 1999; Staab et al. 2000]. In particular, when people are confronted with large amounts of information on a topic they have significant interest in, their goal is to first grasp its overall makeup. They are not interested in details yet, because they have no context in which to interpret them. They need to learn "what is out there" first, so they can begin to build a mental model of the domain into which the details can fit. Of course, often the only way to *do* this is by examining individual instances until themes and trends become apparent. But at least at first, those details are only stepping stones towards a larger objective: finding a way to organize and make sense of the space as a whole.

Considering that this is often the user's goal, it is striking that tools have not been developed to assist this process. Instead, most tools for exploring information spaces are fact-oriented: they aim to draw out the most relevant instances from a larger pool, while leaving to the user the problem of interpreting how those instances are related to one another. Little attention has been given to the larger problem of how to assist humans in discovering the overall patterns and structure in the space. Those attempts that have been made tend to be objective and absolute, rather than encouraging the user to draw his own conclusions and enabling him to materialize and experiment with them.

We believe that computer tools can aid this process by incorporating "mental model" constructs (such as categories) into the interface. The software can then help the user build his conceptual framework, and in a way that is concrete and tangible. We are not claiming any "deep AI" here – this is not empathic software capable of identifying with human perceptions. It simply automates the most tedious parts of the procedure. As we have seen, tools can quickly analyze a human's semantic choices in terms of the underlying data representation, and thereby assist him in recognizing the patterns that he perceived but perhaps couldn't articulate. The next step is to equip him to manipulate those patterns directly, raising the interface experience from the tactical to the strategic. In our future work, we hope to discover that this is indeed a valuable way to help people leverage large amounts of information more naturally and effectively.

References

Grokker 2.1. Sausalito, California, 2004. Available at: www.grokker.com.

Baleydier, L., The KartOO Visual Metasearch Engine. Clermont-Ferrand, France, 2004. Available at: www.kartoo.com.

Boley, D., Gini, M., Gross, R., Han, E.-H., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., and Moore, J., Document categorization and query generation on the World Wide Web using WebACE. *AI Review*, 1999. 13(5-6): p. 365-391.

Chakrabarti, S. and Batterywala, Y. Mining themes from bookmarks. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Boston, Massachusetts. (2000).

Cooley, R., Mobasher, B., and Srivastava, J. Web mining: Information and pattern discovery on the World Wide Web. *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence*. (1997).

Fluit, C., ter Horst, H., van der Meer, J., Sabou, M., and Mika, P., *Spectacle*, in *Towards the Semantic Web: Ontology-Driven Knowledge Management*, J. Davies, et al., Editors. 2003, John Wiley & Sons, Ltd.

Grosky, W. I., Sreenath, D. V., and Fotouhi, F., Emergent semantics and the multimedia semantic web. *SIGMOD Record*, 2002. 31(4): p. 54-58.

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.

Maedche, A., Staab, S., Stojanovic, Studer, R., and Sure, Y. SEAL - A framework for developing SEmantic portALs. *Proceedings of the 18th British National Conference on Databases*. Oxford, UK: Springer-Verlag. (2001), 1-22.

Neches, R., Abhinkar, S., Hu, F., Eleish, R., Ko, I.-Y., Yao, K.-T., Zhu, Q., and Will, P., "Collaborative information space analysis tools," in *D-Lib Magazine*, October 1998.

Peters, R. Exploring the design space for personal information management tools. *Proceedings of the 18th Conference on Human Factors in Computing Systems*. Seattle, Washington. (2001), 413-414.

Santini, S., Gupta, A., and Jain, R., Emergent semantics through interaction in image databases. *IEEE Transactions on Knowledge and Data Engineering*, 2001. 13(3): p. 337-351.

Santini, S. and Jain, R., Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1999. 21(9): p. 871-883.

Shahabi, C. and Chen, Y.-S. Web information personalization: challenges and approaches. *Proceedings of the Third International Workshop on Databases in Networked Information Systems*. Aizu-Wakamatsu, Japan. (2003).

Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Schnurr, H.-P., Studer, R., and Sure, Y., Semantic community Web portals. *Computer Networks*, 2000. 33(1-6): p. 473-491.

Staab, S., Santini, S., Nack, F., Steels, L., and Maedche, A., "Emergent Semantics," in *IEEE Intelligent Systems, Trends and Controversies*, Jan/Feb 2002.

Strehl, A., Ghosh, J., and Mooney, R. Impact of similarity measures on web-page clustering. *Proceedings of the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search*. Austin, Texas. (2000), 58-64.

Wexelblat, A. and Maes, P., Footprints: Visualizing histories for web browsing. MIT: 1997. Available at: http://web.media.mit.edu/~wex/Footprints/footprints1.html.

Yao, K.-T., Neches, R., Ko, I.-Y., Eleish, R., and Abhinkar, S. Synchronous and asynchronous collaborative information space analysis tools. *Proceedings of the International Workshop on Collaboration and Mobile Computing*. Fukushima, Japan. (1999).