Knowledge-Based Navigation of Complex Information Spaces

Robin D. Burke, Kristian J. Hammond & Benjamin C. Young
Artificial Intelligence Laboratory
University of Chicago
1100 E. 58th St., Chicago, IL 60637
{burke, kris, bcy1}@cs.uchicago.edu

Abstract

While the explosion of on-line information has brought new opportunities for finding and using electronic data, it has also brought to the forefront the problem of isolating useful information and making sense of large multi-dimension information spaces. We have built several developed an approach to building data "tour guides," called FINDME systems. These programs know enough about an information space to be able to help a user navigate through it. The user not only comes away with items of useful information but also insights into the structure of the information space itself. In these systems, we have combined ideas of instance-based browsing, structuring retrieval around the critiquing of previously-retrieved examples, and retrieval strategies, knowledge-based heuristics for finding relevant information. We illustrate these techniques with several examples, concentrating especially on the RENTME system, a FINDME system for helping users find suitable rental apartments in the Chicago metropolitan area.

Introduction

Finding items of interest in a large multi-dimensional information space is a problem of growing importance given the ever-increasing amount of information accessible on-line. Standard approaches such as keyword retrieval demand more specificity than the average user can supply. The user must know what he or she wants well enough to create a well-defined query in a query language. The alternative to querying is browsing. Browsing allows users who cannot specify exactly what they seek to rummage around in an information space to find it. However, it is difficult to structure a large browsing space so that users can move about in useful and efficient ways without getting lost.

Our approach to problems of information finding in large multi-dimensional information spaces is to employ *assisted browsing.* The user is provided with standard browsing and/or retrieval interfaces, but his or her progress in this space is monitored and relevant assistance is provided. The aim of assisted browsing is to allow access to information along a multitude of dimensions and from a multitude of sources without the user needing to be aware of these complexities. Since browsing is the central metaphor, we avoid as much as possible forcing users to create a specific queries. At the same time, the intelligent assistance available in the system has the ability to draw in other sources of knowledge. Knowledge-based retrieval strategies can be employed to consider all of the dimensions of the information and present suggestions that lead the user's search in reasonable directions.

The problem

We have implemented our assisted browsing approach in a series of systems called FINDME systems. The class of problems addressed by these systems is best explained through an example:

> You want to rent a video. In particular, you'd like something like *Back to the Future*, which you've seen and liked. How do you go about finding something?
>
> Do you want to see the sequel, *Back to the Future II*? Do you want to see another Michael J. Fox movie? Do you want to see *Crocodile Dundee*, another movie about a person dropped into an unfamiliar setting? *Time After Time*, another time travel film? Another movie by the same director, such as *Who Framed Roger Rabbit?*

The goal of the FINDME project is to develop systems that deal with this sort of search problem. We see this approach as applicable to domains in which there is a large, fixed set of choices and in which the domain is sufficiently complex that users would probably be unable to fully articulate their retrieval criteria. In these kinds of areas, person-to-person interaction also takes the form of trading examples, because people can easily identify what they want when they see it. Many complex selection problems have these characteristics, for example, looking for an apartment. It is difficult to specify completely all of the features you want in an apartment, but it is easier to look at a description and come up with a response such as "I'd like something like this, but a little bigger and in a better neighborhood."

An outline of the solution

Each FINDME systems has two basic parts:

- An initial query or "zooming" (Osgood, 1994) stage through which users state their starting point within the information space, and
- An assisted browsing phase in which users traverse the information space.

User interface is obviously an important part of any FINDME system, particularly in the zooming stage where users enter the system. Examples in this paper will show some of the different interface designs that we have implemented. However, we concentrate in this paper on the mechanisms of traversal through the information space and types of assistance that are provided to users.

Although we have found that different assistance strategies are appropriate in different domains, some categories of intelligent assistance appear relevant in all of our systems:

Trade-off explanation: Users, especially in unfamiliar domains, may fail to understand certain inherent trade-offs in the domain they are exploring. A car buyer might not understand the trade-off between horsepower and fuel efficiency, and attempt to search for a high-powered car that also gets 50 miles to the gallon.

Similarity-based retrieval: As has frequently been found in other information retrieval contexts, it is useful to allow a user to retrieve new items that are similar to an example currently being viewed (Ullman, 1988; Williams, et al. 1982). We found that in most cases overall similarity was a poor metric for providing examples, because users attached different significance to features depending on their goals. If your goal is to buy a car that will pull a big trailer, you will weight engine size more heavily when comparing cars than other features such as passenger leg room.

Tweaking: Browsing is typically driven by differences: if a user were totally satisfied with the particular item being examined he or she would stop there. Often, the item itself can play a useful role in articulating the user's goals. For example, if you are looking for a science fiction movie to rent, you might look at *Terminator II*, but think "That would be good, but it's too violent for my kids." In this case, the task of browsing brings to mind a new feature, level of violence, that now can become an explicit part of further search.

These mechanisms are part of a dialogue between system and user in which the user comes to a better understanding of the domain of examples (through learning about trade-offs and seeing many examples) and the system helps the user find specific items of interest by gradually refining the goal. We also consider it important that none of the choices truly narrow the search: in many retrieval systems, a user who gets off to a bad start can get trapped in a corner of the information space and have to start over. In our systems, we allow the user to redirect the search at any time.

Some FindMe Systems

We have built three FINDME systems for different domains. In general, we have worked from domains with smaller spaces of examples in which features are well-defined and user goals are straightforward, to larger domains with fuzzier features and more complex user goals.

Car Navigator

The first complete FINDME system was the CAR NAVIGATOR, a FINDME system for new cars. Using the interface, which resembles a car magazine, the user flips to the section of the magazine containing the type of car he or she is interested in. Cars are rated against a long list of criteria such as horsepower, price or gas mileage, which are initially set by default for the car class, but can be directly manipulated. Retrieval is performed by turning the page of the magazine, at which point the criteria are turned into a search query and a new set of cars is retrieved. Depending on how the preferences have changed, the system may suggest that the user move to a different class of cars. For example, if the user started with economy cars and started to increase the performance requirement, the system might suggest sports cars instead.

It is possible for the user to put the preferences in an inconsistent state, triggering an explanation of the trade-off that the user has encountered. For example, if a user requests good gas mileage and then requests high horsepower the yellow light will come on next to the gas mileage and horsepower features. The system explains that there is a trade-off between horsepower and gas mileage, and the user will have to alter his or her preferences in order to find any cars.

In addition to the fine-grained manipulation of preferences, CAR NAVIGATOR permits larger jumps in the feature space through buttons that alter many variables at once. If the user wants a car that is "sportier" than the one he is currently examining, this implies a number of changes to the feature set: larger engine, quicker acceleration, and a willingness to pay more, for example. For the most common such search strategies, CAR NAVIGATOR supplies four buttons: *sportier*, *roomier*, *cheaper*, and *classier*. Each button modifies the entire set of search criteria in one step. Although direct manipulation of the features was appealing to users, we found that few users had the patience to twiddle the controls for all of the features. The retrieval strategies were much more effective mechanism for redirecting the search.

Video Navigator

We used our experience in building CAR NAVIGATOR in the construction of a system for browsing the set of movie videos available for rental in a typical video store. This system, VIDEO NAVIGATOR, draws on a database of 7500 movies from a popular video reference work (Wiener, 1993).

The system is organized as a sequence of shelves divided into a variety of categories. The user has several zooming tools that can be used to make an initial query into the shelves. One of these tools is a map of the store organized by standard categories such as the names of actors or directors. Once at a particular shelf, the user can select movies and look at additional information about them, such as plot summary, cast lists, etc.

The retrieval mechanism in VIDEO NAVIGATOR, unlike CAR NAVIGATOR, is separate from the browsing interface. It is implemented in a set of interface agents, called *clerks*. When a user selects a movie to examine, the clerks spring into action. In VIDEO NAVIGATOR, there are four clerks: one recalls movies based on their genre, one recalls movies based on their actors, another on directors, and still another arrives at suggestions by comparing the user against the profiles of other users. Whenever the user picks a movie to inspect, each clerk retrieves and suggests another related movie. It is as if the user has a few knowledgeable movie buffs following her around the store, suggesting movies based on their particular area of expertise.

Since users evaluate similarities between movies differently, we needed to implement a variety of retrieval strategies. However, we opted not to use buttons to implement these strategies. This design choice was due to the nature of the movie domain. Users have seen more movies than they have cars. They know more points in the information space, so need less help from the system in getting around. Instead, we opted to have the strategies operate as independent agents reporting their results to the user. The user can choose to follow up or ignore the suggestions.

RentMe

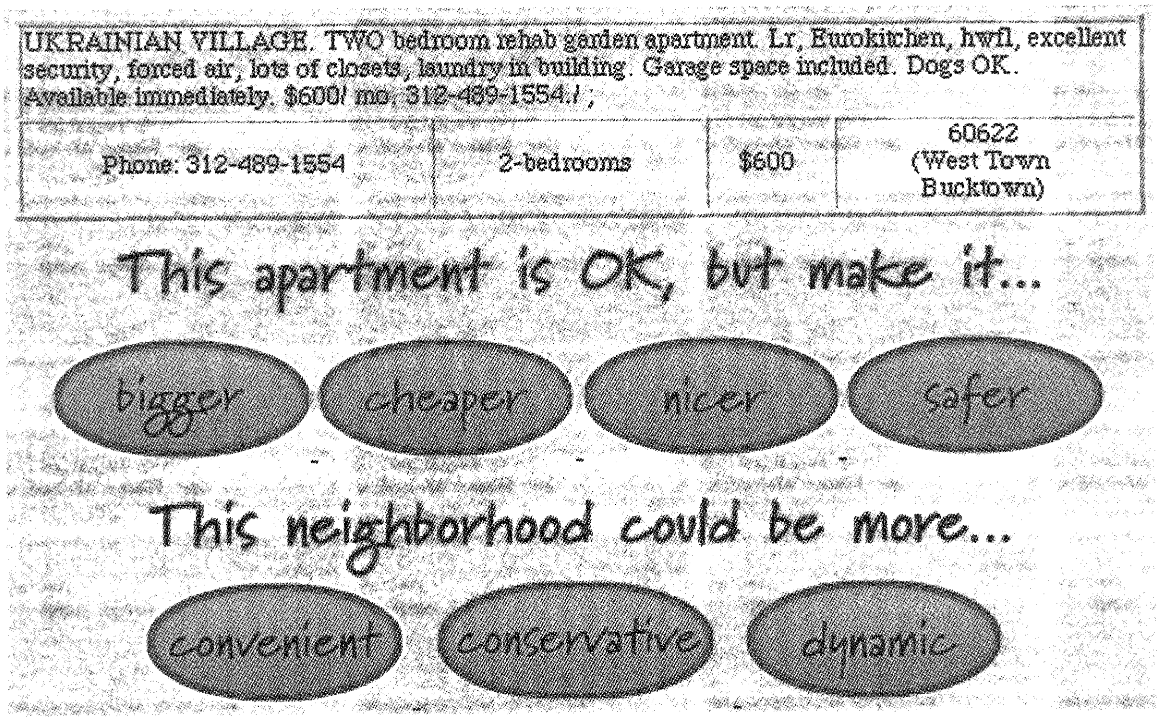
Our newest FINDME system, RENTME, is a World-Wide Web interface to a database of classified ads for rental apartments. In a typical week, thousands of apartments might be advertised for rent in a large metropolitan area. It is straightforward to put this list of rentals in a database and allow searching, but like other domains where FINDME systems have been developed, simple searching is not feasible when users' goals are not expressed in the same vocabulary as the database.

A typical apartment seeker might have a goal like "I'd like a place like what I have now but a little bigger and in a neighborhood with more stuff to do nearby." Notions such as "like the apartment I live in now" are idiosyncratic and can only be evaluated by the person examining a particular apartment listing. It might be difficult or impossible for a user to articulate all of the important features desired in an apartment. Another important aspect of the goal stated above is its reference to knowledge outside of the domain of the apartment listings themselves. To know whether a neighborhood has "more things to do," one must know something about the city itself.

Because of the restrictive nature of the web as a interface, RENTME has a simple interface. We chose to avoid direct manipulation of low-level features (such as those in CAR NAVIGATOR) or interface agents (ala VIDEO NAVIGATOR). The fundamental interaction with RENTME is in the form of critiquing examples using a small set of search redirection strategies like those used in CAR NAVIGATOR. Consider the following example:

The user begins by selecting some general parameters for the search: a particular Chicago neighborhood, 'Bucktown" a price category, $600 a month; and a size, 2 bedroom. The system performs a standard database retrieval and finds a list of apartments that meet the query. This list is typically long and may not contain any apartments that are exactly what the user wants. As shown in Figure 1, the user can improve the search by selecting any apartment and using it as the basis for further retrieval by tweaking it.

The user might, for example, decide that an apartment has the right features, but is simply too expensive. The "Cheaper" button is used to tell the sys-


UKRAINIAN VILLAGE. TWO bedroom rehab garden apartment. Lr, Eurokitchen, hwfl, excellent security, forced air, lots of closets, laundry in building. Garage space included. Dogs OK. Available immediately. $600/ mo, 312-489-1554./ ;

Phone: 312-489-1554	2-bedrooms	$600	60622 (West Town Bucktown)

This apartment is OK, but make it...
bigger
cheaper
nicer
safer
This neighborhood could be more...
convenient
conservative
dynamic

Figure 1: Tweaking an apartment in RENTME

tem to find similar apartments that are cheaper. The system performs another round of retrieval, keeping in mind the features of the apartment the user originally selected. As shown in Figure 2, it only finds one acceptable apartment in the same neighborhood, so it relaxes the neighborhood constraint and begins to look at other, similar, neighborhoods for cheaper apartments.

How RentMe Works

RENTME starts with a database containing the text of apartment ads. There is an initial natural language processing step in which the properties of the apartments are extracted and entered as features in a large database. Then, when users connect to the system, these features are used in the retrieval that follows.

Natural Language Processing

RENTME uses an expectation-based parser (Schank & Riesbeck, 1981) to extract features from the very terse and often-agrammatical language of the classified ads. The system has a large lexicon of terms that are commonly used in ads: in fact, the vocabulary is controlled to a certain degree by the newspaper's style conventions. The recognition of these terms is controlled through expectations that create knowledge structures. Expectation-based parsing makes it possible to distinguish between "No dogs" and "Dogs welcome," a distinction lost to many term-based approaches.

For example, consider the following ad copy:

BUCKTOWN COACHHOUSE. CORTLAND & Hermitage, 1841 N Hermitage, 1 bedroom + den, hardwood floors, eat-in kitchen, yard, laundry. No dogs. $595. 549-5443.

The system represents this as follows:

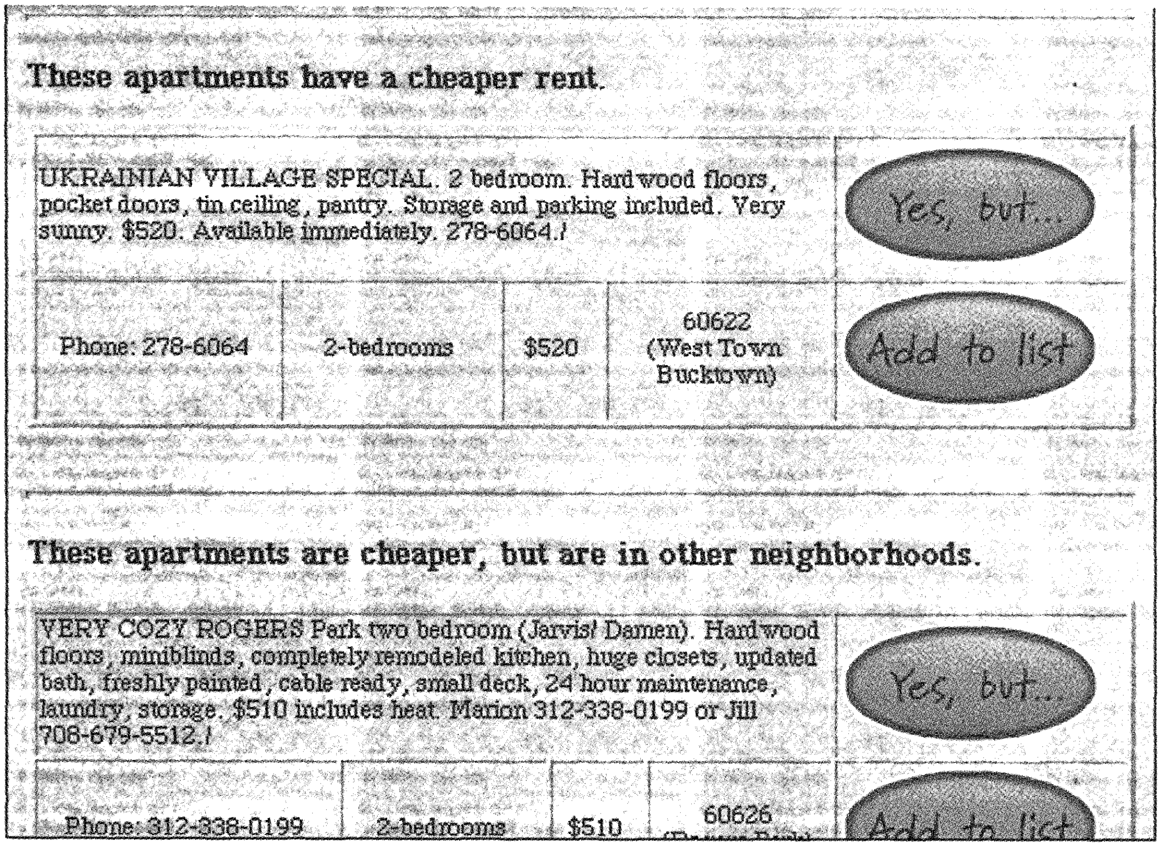
```
(apartment (address "1841 N HERMITAGE")
   (rent 595) (bdr 1) (kitchen eat-in)
   (floor hardwood) (building laundry)
   (pets (dogs (none))))
```

Note that some aspects of the ad are missed, such as the fact that the "apartment" is actually a coach house. We are working to extended the parser to handle more of these cases, but at the moment it extracts about 80% of the important features from ads.

Retrieval Strategies

Central to the system's operation is the *search state* in which the user's current preferences are represented as a set of constraints. For example, the system represents the user's constraints for the initial retrieval in the example above as

```
[600 < price < 650,
```



These apartments have a cheaper rent.

UKRAINIAN VILLAGE SPECIAL. 2 bedroom. Hardwood floors, pocket doors, tin ceiling, pantry. Storage and parking included. Very sunny. $520. Available immediately. 278-6064./

Phone: 278-6064	2-bedrooms	$520	60622 (West Town Bucktown)

These apartments are cheaper, but are in other neighborhoods.

VERY COZY ROGERS Park two bedroom (Jarvis/ Damen). Hardwood floors, miniblinds, completely remodeled kitchen, huge closets, updated bath, freshly painted, cable ready, small deck, 24 hour maintenance, laundry, storage. $510 includes heat. Marion 312-338-0199 or Jill 708-679-5512./

Phone: 312-338-0199	2-bedrooms	$510	60626

Figure 2: The result of applying the "cheaper" tweak

```
    neighborhood = ''Bucktown'',
    size = 2]
```

Each retrieval strategy is represented as a ordered list of operations to be performed on the constraint set. The system only performs as many of these operations as required to retrieve a predetermined number of apartments. The "cheaper" tweak has the following operations:

1. Add a "cheapness" constraint greater than the current apartment, and perform retrieval. This will cause the retriever to look for apartments that have freebies that make them inherently cheaper than the current apartment even though their rent is the same: for example, "heat included."

2. If this method does not retrieve enough candidates, shift the rent constraint to a lower price bracket, and perform retrieval.

3. If there are still not enough candidates, keep the lower rent bracket and decrease the "niceness" constraint if there is one.

4. If more apartments are needed, keep the lower rent bracket and the original "niceness," and alter the neighborhood constraint to a neighborhood that is considered similar to neighborhood where the current apartment is located.

In the example above, no apartments are found with more freebies than the selected apartment. (It already has parking included.) The second method in the strategy finds one apartment that is cheaper in the current neighborhood, but the system is enjoined to return at least 10 candidates, so the other methods in the "Cheaper" strategy are invoked. The user has not yet set a lower bound on how nice the apartment must be, so the third method cannot be used. Finally, the last method is chosen, which identifies a small set of neighborhoods considered similar to Bucktown and looks for similar apartments in those locations.

Knowledge Base

RentMe must have three different kinds of knowledge in order to do its job:

- It must know about the qualities of neighborhoods so it can respond to users' requests for apartments that are nicer, more convenient, etc. since part of the evaluation of an apartment necessarily includes its location.
- It must know about the relative locations of neighborhoods because users who are interested in a particular neighborhood might also be interested in nearby neighborhoods if some particularly good trade-off can be made by redirecting the search there.
- It must know about the features of apartments and how they can be evaluated to arrive at relative levels of niceness, convenience, etc.

In our current version of RENTME, these knowledge types are not as well elaborated as we would like. The qualities of neighborhoods and apartments are given as simple numerical weights, which are readily compared but are highly simplified. For example, apartments with long ads typically score better simply because more features are likely to be listed. To better approximate a human evaluation function, the system needs to take more into account. An apartment that says "Gorgeous apartment in best part of town. $2300 a month." should probably be rated as nicer than an apartment that rents for $550 a month, but whose ad gives a long list of amenities. RENTME needs to represent the different baseline expectations associated with different orders of magnitude in monthly rent.

Related Work

The problem of navigating through complex information spaces is a topic of active interest in the AI community. (See, for example, [Knoblock & Levy, 1995; Burke, 1995].) Much of this research is directed at browsing in unconstrained domains, such as the World-Wide Web, where pages can be on any topic and users' interests are extremely varied. As a result, these systems must use knowledge-poor methods, typically statistical ones.

Our task in FINDME systems is somewhat different. We expect users to have highly-focused goals: such as learning about neighborhoods and finding a suitable apartment to rent. The data being browsed all represents the same type of entity, in the case of RENTME, apartment ads. As a result, we can build substantial, detailed knowledge into our systems that enables them to identify trade-offs, compare entities in the information space, and respond to user goals. All of these properties make FINDME systems more powerful than general-purpose browsing assistants.

In the area of information retrieval, browsing is usually a poor cousin to retrieval, which is seen as the main task in interacting with an information source. The metrics by which information systems are measured do not typically take into account their convenience for browsing. The ability to tailor retrieval by obtaining user response to retrieved items has been implemented in some information retrieval systems through relevance feedback (Salton & McGill, 1983), and through retrieval clustering (Cutting, et al. 1992).

Our approach differs from relevance feedback approaches in both explicitness and flexibility. In VIDEO NAVIGATOR, users see an explicit explanation of why each example was retrieved and can critique particular components of that explanation. In most relevance feedback approaches, the user selects some retrieved documents as being more relevant than others, but does not have any detailed feedback about the features used in the retrieval process. In RENTME, feedback is given through the use of tweaks. The user does not say "Give me more items like this one," the aim of relevance feedback systems, but instead asks for items that are different in some particular way.

Examples have been used as the basis for querying in databases since the development of Query-By-Example (Ullman, 1988). Most full-feature database systems now offer the ability to construct queries in the form of a fictitious database record with certain features fixed and others variable. The RABBIT system (Williams, et al. 1982) took this capacity one step further and allowed retrieval by incremental reformulation, letting the user incorporate parts of retrieved items into the query, successively refining it. Like these systems, FINDME uses examples to help the user elaborate their queries, but it is unique in the use of knowledge-based reformulation to redirect search based on specific user goals.

Another line of research aimed at improving human interaction with databases is the "direct query" approach (Schneiderman, 1994). These system use two-dimensional graphical maps of a data space in which examples are typically represented by points. Queries are created by moving sliders that correspond to features, and the items retrieved by the query are shown as appropriately colored points in the space. This technique has been very effective for two-dimensional data such as maps, but only when the relevant retrieval variables are scalar values representable by sliders.

Like FINDME, direct query approach has the benefit of letting users discover trade-offs in the data because users can watch the pattern of the retrieved data change as values are manipulated. However, direct query systems have no declarative knowledge about trade-offs, and cannot explain to users how they might modify their search or their expectations in light of the trade-off. Also, as we found in CAR NAVIGATOR, direct manipulation is less effective when there are many features to be manipulated, especially when users may not be aware of the relationships between features.

Our use of knowledge-based methods to the retrieval of examples has its closest precedent in retrieval systems used in case-based reasoning (CBR) (Hammond, 1989; Riesbeck & Schank, 1989; Kolodner, 1993). A case-based reasoning system solves new problems by retrieving old problems likely to have similar solutions. Because the retrieval step is critical to the CBR model, researchers in this area have concentrated on developing knowledge-based methods for precise, efficient retrieval of well-represented examples. For some tasks, such as case-based educational systems, where cases serve a variety of purposes, CBR systems use a variety of retrieval strategies that measure similarity in different ways (Burke & Kass, 1995).

Conclusion

FINDME systems perform a needed function in a world of ever-expanding information resources. Each system is an expert on a particular kind of information, extracting information on demand as part of the user's exploration of a complex domain. In FINDME systems, users are an integral part of the knowledge discovery process, elaborating their information needs in the course of interacting with the system. The user need only have general knowledge about the set of items and only an informal knowledge of his needs; he can rely on the system to know about the tradeoffs, category boundaries, and useful search strategies.

Robustness in the face of user uncertainty and ignorance is another important aspect of RENTME and other FINDME systems. Most people's understanding of real world domains such as cars and movies is vague and ill-defined. This makes constructing good queries difficult or impossible. We believe therefore that an information system should always provide the option of examining a "reasonable next piece," of information, given where the user is now. These next pieces are derived through the application of retrieval strategies.

Acknowledgments

The authors would like to thank Dan Kaplan and the Chicago Reader for their contributions to the RENTME project and Tom Weiner for his assistance with VIDEO NAVIGATOR.

References

Burke, R., & Kass, A. 1995. Supporting Learning through Active Retrieval of Video Stories. *Journal of Expert Systems with Applications*, 9(5).

Burke, R. (ed.) 1995. *Working Notes from the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, AAAI Technical Report FS-95-03.

Cutting, D. R.; Pederson, J. O.; Karger, D.; and Tukey, J. W. 1992. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM/SIGIR Conference*, 318-329.

Hammond, K. 1989. *Case-based Planning: Viewing Planning as a Memory Task.* Academic Press. Perspectives in AI Series, Boston, MA.

Knoblock, C. & Levy, A. (eds.) 1995. *Working Notes from the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, AAAI Technical Report SS-95-08.

Kolodner, J. 1993. *Case-based reasoning.* San Mateo, CA: Morgan Kaufmann.

Osgood, R. E. 1994. The Conceptual Indexing of Conversational Hypertext. PhD Thesis, Northwestern University. Issued as Technical Report #52, Institute for the Learning Sciences.

Riesbeck, C., & Schank, R. C. 1989. *Inside Case-Based Reasoning.* Hillsdale, NJ: Lawrence Erlbaum.

Salton, G., & McGill, M. 1983. *Introduction to modern information retrieval.* New York: McGraw-Hill.

Schank, R.C., & Riesbeck, C. 1981. *Inside Computer Understanding: Five Programs with Miniatures.* Hillsdale New Jersey: Lawrence Erlbaum Associates.

Schneiderman, B. 1994. Dynamic Queries: for visual information seeking. *IEEE Software* 11(6): 70-77.

Ullman, J. D. 1988. *Principles of Database and Knowledge-Base Systems Vol 1.* Computer Science Press, 1988.

Wiener, T. 1993.*The Book of Video Lists.* Kansas City: Andrews & McMeel.

Williams, M. D., Tou, F. N., Fikes, R. E., Henderson, T., & Malone, T. 1982. RABBIT: Cognitive, Science in Interface Design. In *Fourth Annual Conference of the Cognitive Science Society.* Ann Arbor, MI: