Published in: 1991 Proceedings of Human Factors in Computing Systems (CHI '91), New Orleans, April–May, 1991

Information Access in Complex, Poorly Structured Information Spaces

Gerhard Fischer & Curt Stevens

Department of Computer Science and
Institute of Cognitive Science
University of Colorado, Boulder

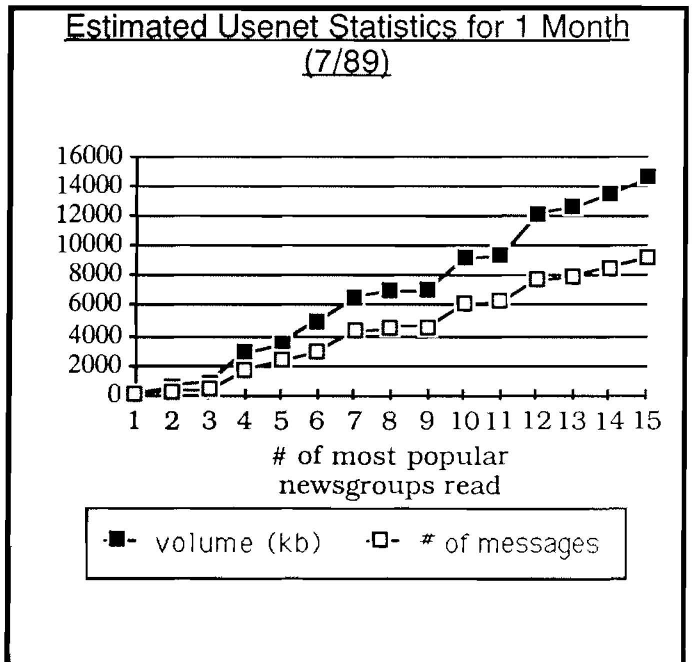
ABSTRACT

Large information spaces present several problems including information overload. This research effort focuses on the domain of Usenet News, an open access computer–based bulletin board system that distributes messages and software. A conceptual framework is developed that shows the need for (a) flexible organization of information access interfaces and (b) personalized structure to deal with vocabulary mismatches. An operational innovative system building effort (INFOSCOPE) instantiates the framework. In INFOSCOPE, users can evolve the predefined system structure to suit their own semantic interpretations. The approach taken by INFOSCOPE differs from other approaches by requiring less up-front structuring by message senders.

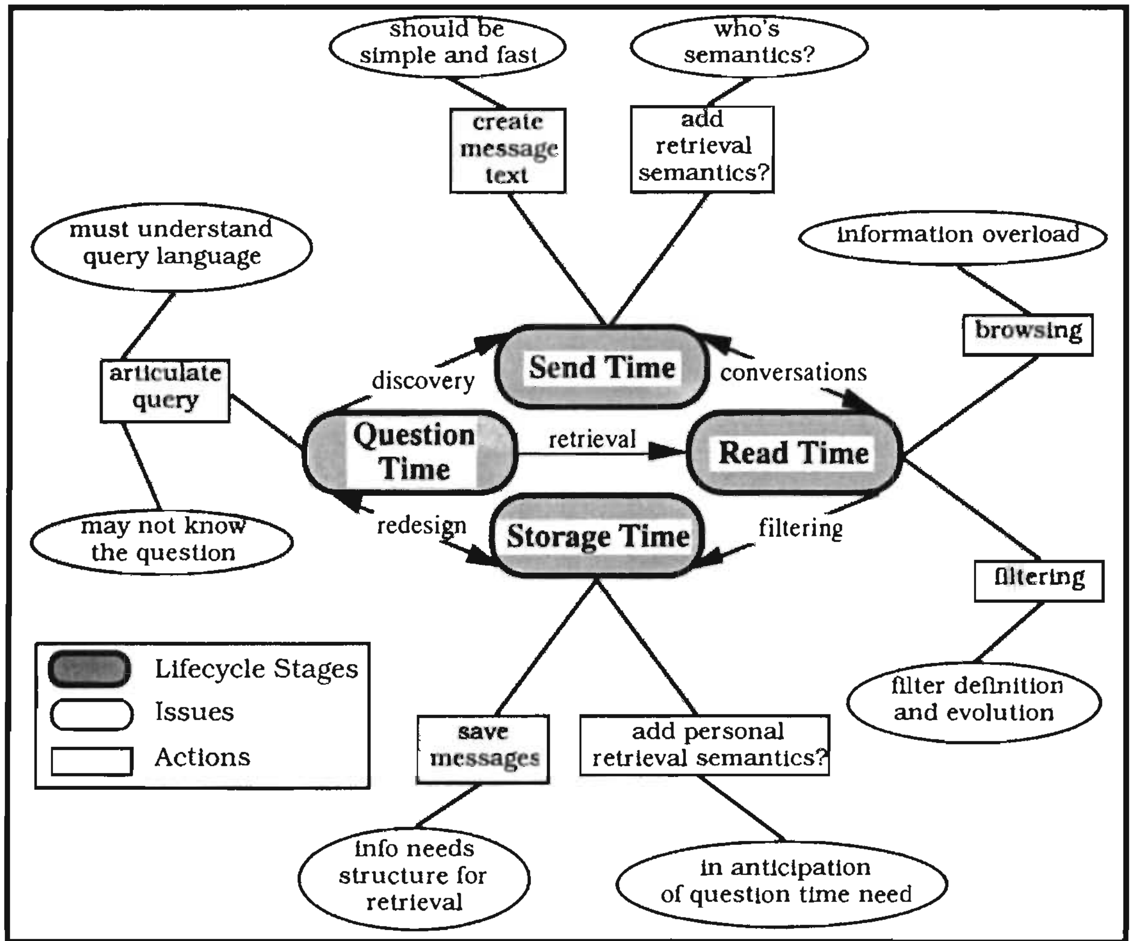
INFORMATION OVERLOAD

Through global networks, the amount of information available to people on their desktops is staggering, leading to an information overload problem [12]. One global network that amply represents this problem is the Internet, especially in its distribution of (Usenet) News. Due to the manner in which News accepts new messages, users are confronted by the following problems: (1) users wishing to distribute messages through Usenet must first classify the message into a specific *newsgroup* and (2) message readers must find messages by browsing an information space with semantic classifications of different message senders. To allow close semantic matches when searching newsgroups, Usenet established a general classification hierarchy for News.

To obtain a deeper understanding of the information overload problem in the context of News, we carried out an empirical study. Our findings demonstrated that reading just a few newsgroups requires perusing megabytes of information each month (see Figure 1). Many of our subjects indicated that they would be interested in reading about more topics, but the large volume of information prohibited them from effectively utilizing the available



1 Usenet News Traffic
Usenet News with its over 400 newsgroups provides a unique opportunity to study information overload. The line marked with with white boxes shows that the 6 most popular newsgroups contain about 3000 messages in a typical month. The total information volume of these 3000 messages is 5000kb (the line marked with black boxes). This graph does not contain the popular source or binaries newsgroups that make the problem much worse.



2 An Information Lifecycle

There are four basic stages in the life of a message. Within each stage there are several tasks and decisions to be made. The arrows between stages in this diagram represent the processes involved in transferring a message from one stage to the next.

information. This suggests that users need help finding relevant information and supports the conclusion that the critical resource is not the available information but the human attention necessary to utilize that information [14]. One hypothesis behind our research is that by reading selected newsgroups more efficiently, users can read more newsgroups.

CONCEPTUAL FRAMEWORK

An Information Lifecycle

Our work in developing systems that help with messages indicates that messages travel through an information lifecycle. As illustrated in Figure 2, when related messages traverse the arc between send time (when messages are created) and read time (when messages are read) a conversation results. However, only some messages are interesting over the long term, and those messages traverse the arc between read time and storage time (when messages are saved for later retrieval). Saving messages indicates that those message topics are interesting.

Once a message is stored, its lifecycle doesn't end. When users have questions (question time) they can access this information. This process of taking a message from question time to read time is called retrieval, and is addressed in our research by the HELGON system [5]. Adding personal retrieval semantics at storage time helps make retrieval easier.

When users have a question but have not previously stored any relevant information, they engage in a process of discovery [13], in which a new message is sent as a request for augmentation of one's own environment. Because the process of discovery involves interaction with many agents (the other users of Usenet News), it is highly unlikely that any two identical queries will result in the same set of answers. This makes the process of discovery quite different from that of retrieval.

The Four Stages Of Information Processing

Each stage represents several actions and issues. Send time processing is the work users do to send a message. It ranges from simply specifying a destination and subject (as in UNIX e-mail), to coarse-grained categorization (as in

Usenet newsgroups), to fine–grained structuring of the semantics of the message content.

Read time processing is the work users do in order to find and read the information of interest. This ranges from simply browsing the information space (like the UNIX RN news reader), to writing specific rules to filter out information that already includes sophisticated semantics.

At storage time users have an opportunity to add retrieval semantics to the message. This ranges from simply saving all messages to a flat file or mbox (like many people use UNIX e–mail), to adding coarse–grained semantics by saving to a folder structure (like some people use UNIX e–mail), to utilizing fine–grained semantics and saving to an area that can be perused using rules.

Question time processing occurs when users need specific information. To retrieve previously stored information users must articulate a query, which ranges from simply searching for text strings, to utilizing coarse–grained semantics (like the UNIX e–mail folders), to utilizing fine–grained semantics stored with the information. If no information is already stored, users must engage in discovery.

Situation and System Models

In addition to the information lifecycle, an analysis of the situation and system models [3] supports this conceptual framework. Users state their desires in the terms of their current task, context or view of the world (the *situation model*), whereas a computer system presents them with an information space organized around someone else's view of the world (the *system model*). For example, a user interested in the language CLOS (common–lisp object system) may be unable to find relevant messages because they reside in the newsgroups comp.lang.lisp and cu.cs.commonloops. There is no CLOS newsgroup, but the user only knows that term (illustrating what Furnas [6] calls the vocabulary problem). Our research investigates various methods for bridging gaps between situation and system models. In most approaches, users receive help in expressing themselves in a system model, forcing them to map their personal situation model to a more global system model. This task consumes valuable cognitive resources. The approach investigated through INFOSCOPE is to design systems that learn about users and allow the expression of structure in the users' own situation models. The system model presented by INFOSCOPE can be refined by users until it more closely matches their situation models.

A similar perspective on this problem, the gulfs of execution and evaluation, has been developed by Norman [11]. The gulf of execution involves the translations from one's goals to the physical system being used to execute those goals. The gulf of evaluation represents the mapping from the physical system, which has completed some portion of the desired task, to the terms and expressions of the goal as viewed by the user.

Another analysis is Moran's external–internal task mapping analysis [10]. In this model the computer system is viewed as "a conceptual world into itself." Moran's analysis shows that in the process of design certain conceptual mismatches arise between the system's internal representations and the external reality of tasks and their specifications in the user's domain of interest.

The Information Lens

This system [9] addressed the overload problem in the domain of electronic mail. There are differences between the news and mail domains. For example, mail message senders must specify a finite list of recipients and usually know who these people are. News message senders, however, know very little or nothing about who or how many readers receive their messages. While this makes the goals of INFOSCOPE and the INFORMATION LENS different, the basic goal of filtering large numbers of messages to find relevant information is shared by both systems. The INFORMATION LENS allows message senders to structure their messages. The message sender selects templates representing a particular message type and header components. Message readers write filters that find messages matching certain structures. For example, a student might write a filter to collect all "meeting announcement" message types that contain a "professor" type in the "From:" header line. This idea works well in a small environment of users where standards and expectations can be imposed, but it may not work well in a global environment. It may also fail in environments where people just want to get their work done [2]. The reasons for this relate to the cost/benefit ratio as perceived by the sender of messages [7]. The sender is not the reader, and because adding structure helps the reader find the message, there is no direct benefit to the sender by adding that structure. Typical senders merely wish to write the text of a message and send it. Because senders are often not willing to provide extra structure [8], the INFORMATION LENS allows users to define filters without templates, using the text contained in header fields. Unfortunately, this apparently bypasses the semi–structured message templates already described and may force users to define two sets of filters. The first set describes filters requiring template components and the second describes those using plain text since some messages won't contain template components.

Our research is based upon the hypothesis that senders will not expend the extra cognitive effort necessary to carefully classify messages, but readers will expend a limited amount of effort in restructuring because they perceive a direct benefit (messages are easier to find) from doing so. This is done by allowing readers to restructure the information space according to individual semantics when messages are read.

INFOSCOPE

The INFOSCOPE project includes a system implementation encompassing many of the principles discussed earlier. This paper describes three specific parts: (1) a graphically based user interface for accessing News messages, (2) *virtual newsgroups*, a method for spanning the gaps between the situation and system models in the News domain, and (3) *agents* that address the problems created by implementing virtual newsgroups.

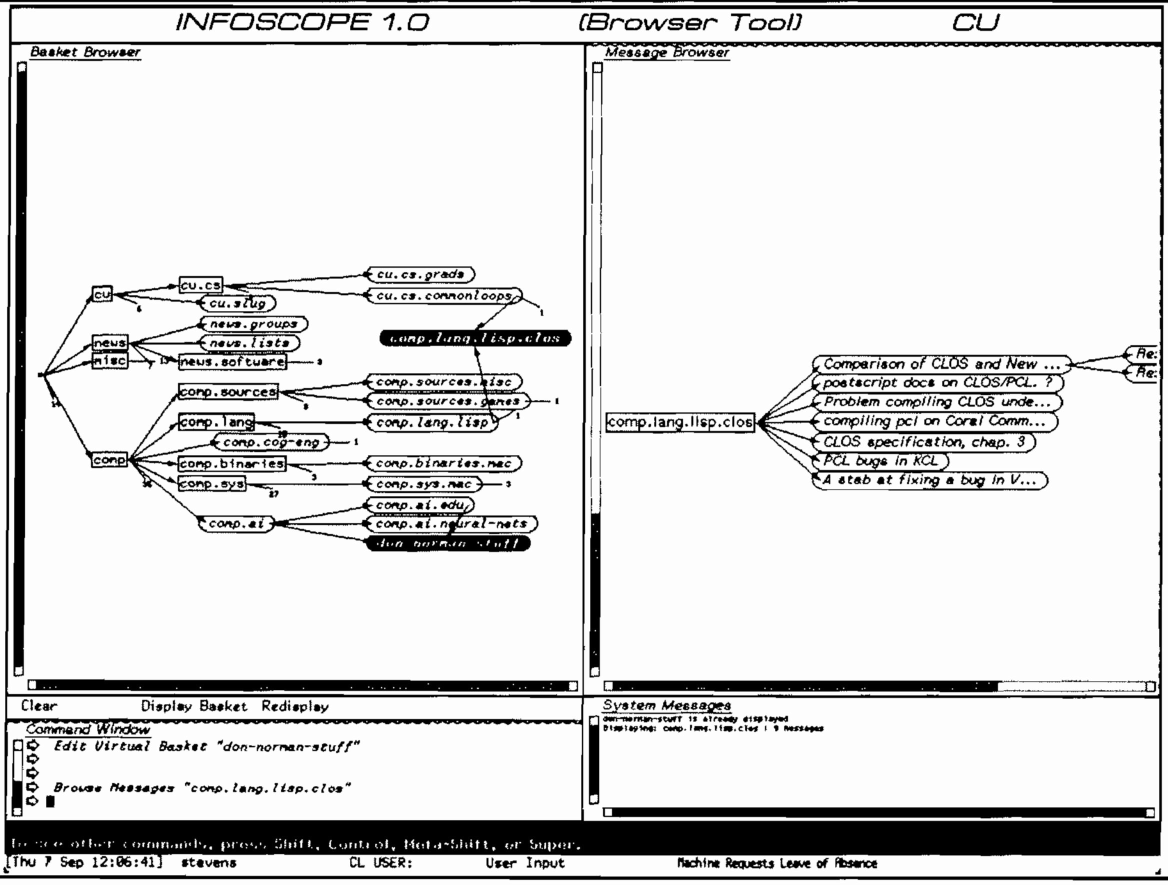
Infoscope vs. Text–Based Interfaces

Several interfaces to the News information space exist. The most common is called RN. It presents users with a succession of text lines containing the names and status of newsgroups (see Figure 3).

The RN Interface to Newsgroups

```
***   1 unread article  in alt.hypertext--read now? [ynq]
***  13 unread articles in alt.sources--read now? [ynq]
***   1 unread article  in boulder.general--read now? [ynq]
***   2 unread articles in co.general--read now? [ynq]
***  23 unread articles in comp.ai--read now? [ynq]
***  14 unread articles in comp.binaries.mac--read now? [ynq]
***   9 unread articles in comp.cog-eng--read now? [ynq]
***  99 unread articles in comp.lang.lisp--read now? [ynq]
***   1 unread article  in comp.mail.headers--read now? [ynq]
***   6 unread articles in comp.newprod--read now? [ynq]
***  11 unread articles in comp.sources.games--read now? [ynq]
***  30 unread articles in comp.sources.misc--read now? [ynq]
***   1 unread article  in comp.sources.unix--read now? [ynq]
***   1 unread article  in comp.sys.mac.digest--read now? [ynq]
*** 289 unread articles in comp.sys.mac--read now? [ynq]
```

3 Text–Based Newsgroup Interfaces
Text–Based interfaces do not display the relationships between newsgroups in the information space. Browsing in this type on environment can be an arduous task.



```
Edit Virtual Basket "don-norman-stuff"
Browse Messages "comp.lang.lisp.clos"
```

```
don-norman-stuff is already displayed
Displaying: comp.lang.lisp.clos | 9 Messages
```

```
To see other commands, press Shift, Control, Meta-Shift, or Super.
[Thu 7 Sep 12:06:41]   stevens        CL USER:        User Input        Machine Requests Leave of Absence
```

4 Browser Tool
The Browser Tool allows users to peruse and learn the existing structure of baskets in INFOSCOPE. Messages within a basket are also browsed in this tool. From here the user can click the mouse on objects to invoke reading, posting, or virtual structure operations. Rectangular nodes are intermediate classifications and are associated only with basic actions such as "display some children." Oval nodes contain messages and represent Usenet newsgroups. These are associated with operations such as "browse messages," and "create virtual basket." The highlighted oval nodes are user–defined virtual newsgroups and represent extensions to the Usenet structure.

An initialization file orders the presentation of newsgroups, but RN provides no tools for managing this file. Although this type of interface might be fine for users who are experienced with the software and familiar with the organization of the information space, it does not illustrate this structure in an effective manner. The INFOSCOPE user interface is a graphical one, which represents the organization of the News information space by displaying the newsgroup hierarchy on the computer screen.

The INFOSCOPE basket browser contains classifications, newsgroups, or virtual newsgroups which requires three node representations. In Figure 4 the *comp* and *comp.lang* nodes serve to illustrate the structure of the information space. The *comp.lang.lisp* node is a Usenet newsgroup and the *comp.lang.lisp.clos* node is a virtual basket containing selected messages filtered from the newsgroups *comp.lang.lisp* and *cu.cs.commonloops*. Virtual newsgroups represent the areas of keenest interest to the individual user and will become the users' center of attention. This reorganization of the information space helps narrow the situation/system model gap by presenting classifications based in user terminologies.

Messages

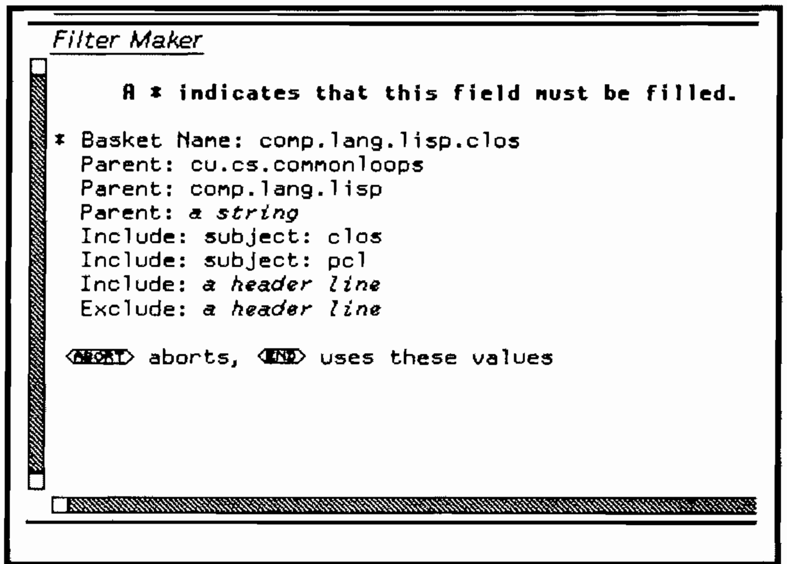
In addition to browsing and manipulating the structure of baskets in INFOSCOPE, users can use the message browser pane of the browser tool to display and navigate through the set of messages contained in a basket. The message browser on the right side of Figure 4 shows messages from the virtual basket *comp.lang.lisp.clos*.

In RN software, the lack of structure imposed on logical collections of messages confounds many novice users. Conversations are often more than "one–shot" dialogs. When several users post a response to a particular message, those messages may each spawn subconversations or *threads*. RN displays messages in the chronological order in which they arrive at an individual's computer. Although the system provides a command for scanning all messages with the same subject, the structure of threads is lost in the ordering. INFOSCOPE organizes messages into conversations and threads. A conversation consists of a message node, all responses to the message in that node, and similarly all responses to the responses until leaf nodes are reached. This idea is demonstrated in Figure 4 by the conversation about "Comparison of CLOS and New Flavors." The two partially displayed messages starting with "Re:", and attached to the "Comparison..." node, are responses to the original posting. "Re:" stands for 'regarding' and is placed in the subject line of a message by the standard news software whenever a reply is posted. Messages in response to a response will have "Re: Re:" at the beginning of the subject line and so on as the conversation flourishes.

There are several advantages to displaying the structure of messages in this manner. In the INFOSCOPE system conversations may be treated as an object. This means that operations such as "Save this conversation," can be used to write all of the messages of a conversation to a file at once. Saving subconversations is as simple as saving entire conversations. For example, if someone responds to a posting with a particularly interesting comment, a whole new conversation may start within a conversation. This conversation can be saved by clicking the mouse on its root node.

Virtual Newsgroups

In order to browse virtual baskets, they must first be created. This process takes place in the basket filter tool (see Figure 5). To create virtual structure, users choose one of the parents of the virtual basket, which can be either real or virtual baskets. When users select either "Create Virtual Basket" or "Edit Virtual Basket" they are taken into the filter tool and presented with a partially filled out filter. INFOSCOPE uses whatever information it has to fill in the filter. When creating a new basket, it fills in a default name for the new basket and the parent that was used to enter the restructuring process. In the example of Figure 5, users are editing the definition for the virtual basket comp.lang.lisp.clos. Using this new virtual basket, users have a repository for information about CLOS that is displayed using the name semantically attached to that information by the user who defined the basket. By



Filter Maker

A * indicates that this field must be filled.

```
* Basket Nane: comp.lang.lisp.clos
  Parent: cu.cs.commonloops
  Parent: comp.lang.lisp
  Parent: a string
  Include: subject: clos
  Include: subject: pcl
  Include: a header line
  Exclude: a header line

 <ABORT> aborts, <END> uses these values
```

5 Defining Virtual Structure
The Filter Maker allows users to create and modify the filter descriptions for virtual baskets. In this example, the virtual basket *comp.lang.lisp.clos* is being edited. The basket inherits messages that contain the subjects **pcl** or **clos** from the existing baskets *comp.lang.lisp* and *cu.cs.commonloops.*

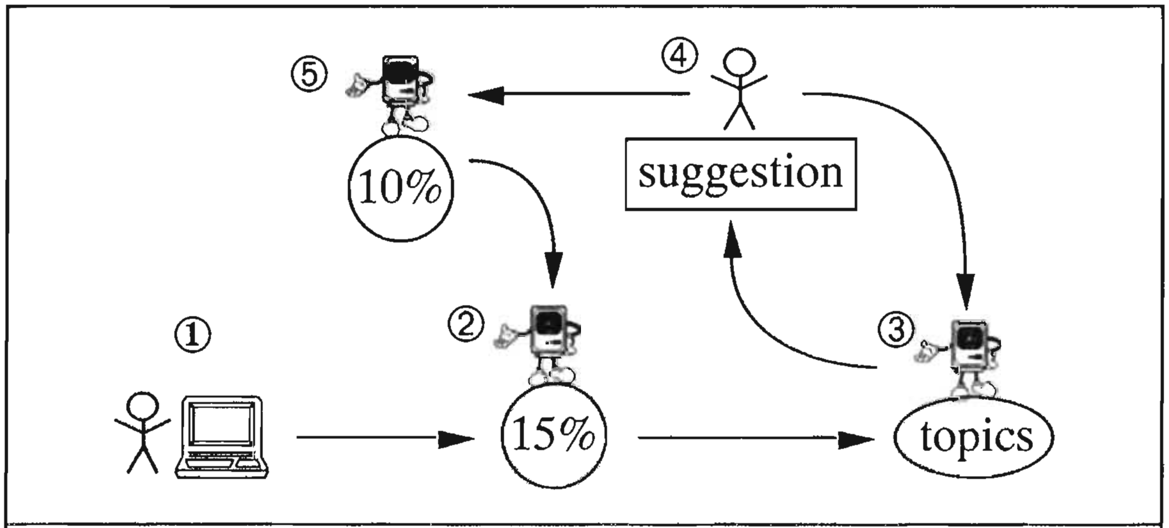
allowing individual users to define virtual baskets, one global system model can be translated into individual situation models.

Agents

Unfortunately, this kind of system model personalization, like any structure modification task, is very difficult. The structure must be based on an analysis of user behavior patterns of which users are not aware. A computer system can assist users in analyzing this behavior based on certain concepts of how users act, and can be used to assist structure creation and modification. In INFOSCOPE, these methods are called agents.

INFOSCOPE agents are collections of rule-based heuristics that utilize information resulting from the analysis of user behavior to make suggestions to the user. These suggestions concern the future modification of the system model that the agents deem necessary for proper evolution of the structure. Users can then accept, reject, or modify these suggestions, and evolution proceeds in this cooperative fashion (see Figure 6). In addition, the agents analyze suggestions that are rejected or modified so that the user models can be appropriately adjusted. When a suggestion is rejected, something should be done to ensure that the same error in judgement is not included in future suggestions. The agents that do this analysis of suggestions and modification of other agents are called supervisory agents. They do not modify *what* the agents do, but why and when they do it. From the user's perspective, agents are a feedback mechanism, helping the user modify the displayed structure based on their own system usage patterns. Agents post suggestions to the user that must be accepted to be implemented. In INFOSCOPE, agents help construct virtual filters and users critique those suggestions, making them more accurate. This is easier for users than forcing them to recall enough information to define useful filters.

The mechanisms that agents use to carry out many of their functions are loosely based on Anderson's Rational Analysis of Human Memory [1]. His analysis leads to the definition of several effects that help measure how effectively the history of usage patterns predicts current usage patterns, or the probability that an item is needed given the history of need for such items. For this reason agents keep track of numerous pieces of information about system activities. The three effects that Anderson discusses are frequency, recency, and spacing. Frequency refers to the number of times an item is needed within a specified period of time. Recency refers to the amount of time elapsed since the last need for an item, and spacing refers to distribution across time of the exposures to these items. By holding certain factors constant in specific situations, equations (power functions) are developed that serve as predictors of the current need for an item that has been needed in the past. For example, an item that has been needed 10 times in the last month is more likely to be needed now if its past exposures are distributed evenly across the time span than if all of the past instances occurred in the first three days of that month. Similarly, if an item was needed once a long time ago the probability of current need is less than that for an item that was used very recently. These effects require agents to decay the value of message topics over time when their frequency is low,



6 A Group of Agents

At ①, the user interacts with INFOSCOPE for a period of time that extends until agent ② recognizes that the user consistently reads less than 15% of the available messages in a particular newsgroup. At this time agent ② tells agent ③ to begin gathering topics that are judged interesting and uninteresting by the user. When a sufficient volume of data has been gathered, agent ③ posts a suggestion to the user. (④) If the user accepts the suggestion agent ③ continues gathering topics for modifying filters in the future. If the user rejects the suggestion, agent ⑤ tries to find out why and uses this information to modify agent ②. In this example the user complains that the agent is too intrusive. It is modified so that it only reacts when the user reads 10% or less of the messages in a newsgroup.

increase them when the recency is high, temper those evaluations with analysis of the spacing effects and so on. In INFOSCOPE, users get direct input into the analysis in breakdown situations. If the system determines something to be of interest and the user disagrees, a dialog ensues in order to clear up the discrepancy.

INFOSCOPE agents monitor two general areas of system usage. The first of these deals with the user's individual preferences in daily operation of the system. This includes which baskets should be initially displayed by default and many other aspects of day–to–day operation. The second aspect of system usage that agents will monitor is the actual content of messages that are deemed *interesting* or *uninteresting*. This task is a statistical one, requiring agents to keep track of what users are reading and what they are ignoring over long periods of time. This happens on two levels. At the higher level the system monitors the amount of interesting information found on a regular basis within a specific basket. When the amount is too low, agents suggest modifying the existing structure. At the lower level, the topics covered by a message are retained in order to build and evolve filters. From these two parts of the user profile, INFOSCOPE agents learn about individuals who use the system regularly.

DISCUSSION AND FUTURE WORK

The information overload problems created by large information repositories make it difficult for users to find relevant information. Database query languages are only of use in highly structured and homogeneous information spaces. Usenet News is mostly explored with browsing techniques. Another way to address the problem is allowing users to send semi–structured messages. Unfortunately, there are indications that users are unwilling to put forth the necessary effort [8]. A new approach to this problem is helping users design more effective filters dynamically at read time. This shifts the burden of the extra structuring to the person who most benefits from this effort.

INFOSCOPE is a fully operational prototype implemented on Symbolics Lisp machines. It eliminates a number of drawbacks of RN as an environment to access large, poorly structured information spaces. It replaces RN's teletype-oriented dialog with a state–of–the–art interface incorporating modern user interface tools. It provides (1) a global view of the information space, (2) support for conversations, (3) filtering methods for personalization, and (4) agents assisting in analyzing user behavior. An important aspect of INFOSCOPE is that it operates on top of the Usenet system.

INFOSCOPE can also be viewed as an extension to parts of the INFORMATION LENS [9]. While INFOSCOPE does not include a set of template types, it augments the users' ability to define filters by introducing agents that help create more effective filters. The techniques used can extend systems like the INFORMATION LENS because those systems require non–trivial amounts of effort on the part of message readers.

Limitations

INFOSCOPE intentionally does not make any effort to analyze the content of messages. Currently the system analyses the content of the header fields of a message. Because vocabulary in the header field may not match the vocabulary of the message body, relevant messages may be missed with filters. The framework behind INFOSCOPE allows the addition of semantic knowledge and constraints incrementally, but our current effort has not focused on this direction.

INFOSCOPE's ability to define filters is currently limited to a small subset of boolean logic, which limits the expressiveness of the filters that can be constructed. For example, a filter cannot target only messages with all of the words Lisp, CLOS, and efficiency. It will find all messages with any subset of those words.

Filters represent personal structure on top of messages, allowing structure to be saved with the messages. In order to support the full information lifecycle (see Figure 2), INFOSCOPE should be more tightly integrated with HELGON. An integration effort that joins a retrieval system that can utilize this structure, like HELGON [5], would increase the power of both systems.

Future Work

An important next step is conducting a longitudinal study in a naturalistic setting. We will do this first with a sizable user community at our university. This initial study will be carefully analyzed, and enhancements will be incorporated in the system before distribution to a number of other sites that have already indicated their interest in using the system. We have defined a number of objectives to be carefully monitored in the study, such as how much virtual structure users of INFOSCOPE will be willing to provide and whether the system will reduce information overload by helping users focus on their interests.

Although we hope that agents will provide users with relevant information for the construction of filters, we will replace the current filter–making tool with a filter design environment [4].

ACKNOWLEDGMENTS

We wish to thank the other members of our research project, especially Evelyn Ferstl, Peter Foltz, Scott Henninger, Walter Kintsch, and David Redmiles with whom we shared many discussions about the ideas and the systems discussed in this paper. This research was supported by Grant No. MDA903–86–CO143 from the Army Research Institute.

REFERENCES

[1] Anderson, J. R.; *The Adaptive Character of Thought*; Lawrence Erlbaum Associates: Hillsdale, New Jersey, 1990.

[2] Carroll, J. M.; Rosson, M. B.; Paradox of the Active User; In *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*; J. M. Carroll, Ed.; The MIT Press: 1987; pp 80-111.

[3] Dijk, T. A. v.; Kintsch, W.; *Strategies of Discourse Comprehension*; Academic Press: New York, 1983.

[4] Fischer, G.; McCall, R.; Morch, A.; Design Environments for Constructive and Argumentative Design; In *Human Factors in Computing Systems, CHI'89 Conference Proceedings (Austin, TX)*; ACM: New York, 1989; pp 269-275.

[5] Fischer, G.; Nieper-Lemke, H.; HELGON: Extending the Retrieval by Reformulation Paradigm; In *Human Factors in Computing Systems, CHI'89 Conference Proceedings (Austin, TX)*; ACM: New York, 1989; pp 357-362.

[6] Furnas, G. W.; Landauer, T. K.; Gomez, L. M.; Dumais, S. T.; The Vocabulary Problem in Human-System Communication; In *Communications of the ACM* 1987, *30*, 964-971.

[7] Grudin, J.; Why CSCW Applications Fail: Problems in the Design and Evaluation of Organizational Interfaces; In *Office: Technology and People* 1989, *4*, 245–264.

[8] Mackay, W. E.; Malone, T. W.; Crowston, K.; Rao, R.; Rosenblitt, D.; Card, S. K.; How Do Experienced Information Lens Users Use Rules?; In *Human Factors in Computing Systems, CHI'89 Conference Proceedings (Austin, TX)*; ACM: New York, 1989; pp 211–216.

[9] Malone, T. W.; Grant, K. R.; Turbak, F. A.; The Information Lens: An Intelligent System for Information Sharing in Organizations; In *Human Factors in Computing Systems, CHI'86 Conference Proceedings (Boston, MA)*; ACM: New York, 1986; pp 1-8.

[10] Moran, T. P.; Getting into a System: External-Internal Task Mapping Analysis; In *Human Factors in Computing Systems, CHI'83 Conference Proceedings (Boston, MA)*; ACM: New York, 1983; pp 45-49.

[11] Norman, D. A.; Cognitive Engineering; In *User Centered System Design, New Perspectives on Human-Computer Interaction*; S. W. D. D.A. Norman, Ed.; Lawrence Erlbaum Associates: Hillsdale, NJ, 1986; pp 31-62.

[12] Reddy, R.; Technologies for Learning; In *Computers in Education: Realizing the Potential, Report of a Research Conference*; U.S. Government Printing Office: Washington, D.C., 1983; pp 49-60.

[13] Schwartz, M. F.; The Networked Resource Discovery Project; In *Proceedings of the IFIP XI World Congress*; San Francisco, California, 1989; pp 827-832.

[14] Simon, H. A.; *The Sciences of the Artificial*; The MIT Press: Cambridge, MA, 1981.