Building Task-Specific Interfaces to High Volume Conversational Data

Loren G. Terveen, William C. Hill, Brian Amento, David McDonald, Josh Creter
AT&T Labs - Research
600 Mountain Avenue
Murray Hill, NJ 07974
+1 908 582-2608
{terveen, willhill, s_brian, s_david, s_creter}@research.att.com

ABSTRACT

As people participate in the thousands of global conversations that comprise Usenet news, one thing they do is post their opinions of web resources. Phoaks is a collaborative filtering system that continuously parses, classifies, abstracts and tallies those opinions. About 3,500 users per day consult Phoaks web pages that reflect the results. Phoaks also features a general architecture for building similar collaborative filtering interfaces to conversational data. We report here on the Phoaks resource recommendation interface, the architecture, and the issues and experience that make up its rationale.

Keywords

human-computer interaction, human interface, computer-supported cooperative work, organizational computing, social filtering, collaborative filtering, data mining, resource discovery, World Wide Web, Usenet, Netnews.

INTRODUCTION

Global online conversations, such as Usenet newsgroups and public or corporate mailing lists, contain significant information. However, the useful information may be thinly scattered over hundreds or thousands of high volume message streams. A favored analogy for complaining about this situation is to speak of a *low signal to noise ratio.*

The goal of our research is to create and understand task-specific interfaces to high volume public and corporate conversations. The Phoaks (People Helping One Another Know Stuff) experimental system is our vehicle for doing this research.

The first Phoaks application attacks the problem of mining recommendations of Web resources (URLs) from Usenet newsgroups. Mining URL recommendations from conversations is an interesting problem because finding high quality, topically relevant URLs is difficult, and because it is plausible that people participating in an online conversation about a topic will exchange opinions about online resources for that topic. Usenet is an interesting and challenging research arena because of its scale (1 GB per day), fast turnover, lack of structure (i.e., message bodies are free form text), and the wildly varying quality of information.

In pursuit of our research goal, we are committed to a field research methodology. We believe that only by confronting the scale and messiness of the real world (both in the conversational data we process and the international user community for the Web interface we generate) can we discover the important problems, let alone begin to craft solutions. Therefore, Phoaks processes about 100,000 netnews messages in more than 1500 newsgroups each day, looking for mentions of web resources (URLs). Phoaks maintains an experimental web site (http://www.phoaks.com/phoaks/) that contains over 37,000 pages of recommendation data for these newsgroups. The web site went live in February 1996. As of the end of December 1996, it had been accessed by about 300,000 users from 100 nations, and we have received more than 600 feedback messages. The utility of the feedback and access data has confirmed the value of our methodology. We have made dozens of changes to our system and have learned interesting general lessons about creating interfaces to conversational data. Phoaks has taught us all we expected to learn and much more [13,21].

One of the primary results of our experience was that we generalized the system architecture, moving beyond a "one of a kind" system toward a general framework for creating interfaces to conversational data. We realized support for rapid development, analysis, and continuous modification was crucial. Further, messages from people interested in collaborating with us and our own continuing investigation made us aware of many problems similar to that of mining recommended web resources from netnews. Examples include recognizing FAQ postings (Frequently Asked Questions), pointers to educational resources recommended by expert teachers, commercial brands,

product references, stock symbols. Therefore, we redesigned the Phoaks architecture as a generic engine for mining, processing and presenting conversational data that can be configured for specific tasks. We also evolved a set of tools to aid designers in creating, analyzing, and iterating the system and interface design.

In the remainder of the paper, we first articulate the principles of the Phoaks approach and discuss how they relate to other research efforts; we then describe the Phoaks URL recommendation interface, illustrating our principles for collaborative filtering interfaces; next we describe the system architecture, motivating its design features in terms of our experience and principles; finally, we discuss general lessons we have learned about interfaces to conversational data from Phoaks users.

PHOAKS PRINCIPLES AND RELATED WORK

Traditional message filtering systems, such as SIFT [25], GroupLens [18], URN [3] and Infoscope [6, 20], recommend a subset of messages they judge human readers will find relevant. Phoaks is in a different tradition, that of computer-mediated collaborative filtering systems [14]; its job is to recognize, manipulate, store and present content that occurs first in on-line conversational streams. Three design principles distinguish it from other computer-mediated collaborative filtering systems: *role specialization, reuse and recontextualization.* We expect that these principles will be useful for future Phoaks-like collaborative filtering systems.

Many collaborative filtering systems, particularly ratings-based systems [2, 12, 18, 19] are built upon the assumption of *role uniformity.* They expect all users to do the same types of work and share the same types of benefits: for example, in the case of ratings-based systems, everyone rates objects of interest. Yet there is evidence that people naturally prefer to play different roles in the information ecology [15]; in particular, only a minority of people expend the effort of judging information and volunteering their opinions to others. Independently, we have observed such role specialization in Netnews [21]; authors volunteer long lists of recommended web resources at a stable but extremely low rate. Phoaks assumes that the roles of recommendation provider and recommendation recipient are specialized and different.

Phoaks *reuses* recommendations from existing online conversations. This reuse requires no extra work from providers and no judgments of information quality from Phoaks users. What qualities make for successful reuse of conversational data? Whittaker [22] evaluated the factors that affect use of Lotus Notes™ as an organizational memory. Despite users' intuitions in favor of small homogenous project team use, he found that large numbers of diverse participants and database size correlated with active use of the message databases. The Phoaks approach is aligned with this finding. Other systems are exploring reuse of conversational data. Recently, Marx and Schmandt [16] reported on a personalized and dynamic system (CLUE) that selects personal email message content for reuse in computational support for ongoing daily activities. Ackerman and McDonald [1] describe a new generation of Answer Garden technology which promotes the reuse of corporate knowledge for technical and other kinds of support. Hammond et al.'s FAQfinder system [9] examines the role that evolutionary programming can play in computer-mediated reuse of FAQ-style information.

Reusing recommendations leads to a type of *virtual* collaboration between the producers and consumers of the recommendations. This computer-mediated collaboration offers some of the benefits of collaboration — e.g., efficiently gaining from the experience and expertise of others — without requiring social relationships or explicit communication. However, users of collaborative filtering systems often want to communicate to develop further the mutual interests they have discovered [12]. Apropos this desire and the possibilities inherent in the recommendation mining style of collaborative filtering, the Phoaks design strikes a balance between participants' needs for privacy and desires for connectedness.

The third design principle that Phoaks follows is *recontextualization.* Reusing information for new purposes forces either decontextualization or recontextualization of the information. Like Phoaks, the Green Eggs Report (www.ar.com/ger/index.html — now defunct) and the Link News web site (www.toriwaki.nuie.nagoya-u.ac.jp/~fujii/Link/News/) present web pointers that occur in Usenet newsgroups. However, these systems present the pointers as decontextualized lists. In contrast, Phoaks pays a great deal of attention to context and builds the human interface upon it [10, 11, 23, 24]. As we discuss in detail below, Phoaks selects and orders URLs to present by computing over contextual information and includes selected contextual information as a resource for evaluating recommended URLs.

PHOAKS URL RECOMMENDATION INTERFACE

The first Phoaks application attacks the problem of extracting recommendations of Web resources (URLs) from Usenet messages and creating interfaces to the recommendation data. It builds on work [13, 21] that provides empirical evidence that Usenet messages are a useful source of resource recommendations. To summarize these results:

- Usenet messages often mention pointers (URLs) to Web resources.
- A significant portion of resource mentions are done for the purpose of recommending a resource.

- Recommendation instances can be recognized accurately by machine.
- Some resource recommendations are confirmed by more than one person.
- The number of distinct recommenders of a resource is a plausible measure of resource quality.

Figure 1 illustrates the final result. It shows the intersection between resources recommended on Usenet (that were not in FAQ messages) and resources in newsgroup FAQs for 313 newsgroups. Since FAQs contain information that a human topic expert thinks is of appropriate quality and relevance, they are an appropriate baseline for judging the quality of resources that Phoaks classifies as recommendations. The X axis of the graph in Figure 1 shows resources (mined by Phoaks) as ranked from 1 to 20 by the number-of-recommenders measure, and the Y axis shows the percentage of resources from each rank that are present in the related FAQ(s). For example, 29% of the 313 top ranked resources, 22% of the 313 second ranked resources, and 19% of the 313 third ranked resources occur in the relevant FAQ(s). The graph shows that the more distinct recommenders a resource has, the more likely it is to appear in the FAQ. Thus, the number-of-recommenders measure is consistent with human judgments of quality.

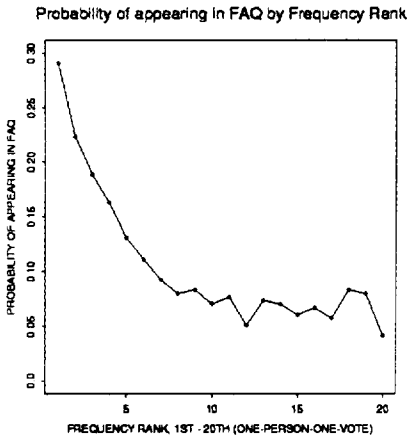

Figure 1: Recommended resources vs. FAQ resources

Phoaks searches netnews for mentions of URLs, finding about 39K each day. It applies rules that categorize each mention (the rules have an accuracy of nearly 90%; see below for details). For our purposes, the most important categories are recommendations and contact pages, since these are the resources we want to present in our interface. Phoaks maintains a database of recommended resources and associated contextual information, and generates web pages as an interface to the recommendation data.

The Phoaks Web Site

The Phoaks web site contains information for about 1500 newsgroups (some of the groups we monitor turned out to be inactive; a few others have some activity but don't appear to mention URLs). The interface presents a tightly interlinked set of pages. The major page types are:

- *resource summaries* — information about resources, number of recommenders for each resource, and the frequency and recency of recommendation.
- *recommender summaries* — frequent recommenders and the resources they recommended.
- *resource pages* — people who recommended a particular resource.
- *recommender pages* — resources recommended by a particular person.
- *message pages* — contextual information from a message in which a resource was recommended.
- *index pages* — summaries of "internal nodes" in the news hierarchy (like rec.music), used for navigation.
- *feedback pages* — users can add links to the site, give opinions on links already present, and express comments about Phoaks.

Figure 2 shows the resource summary page for the newsgroup rec.music.dylan.

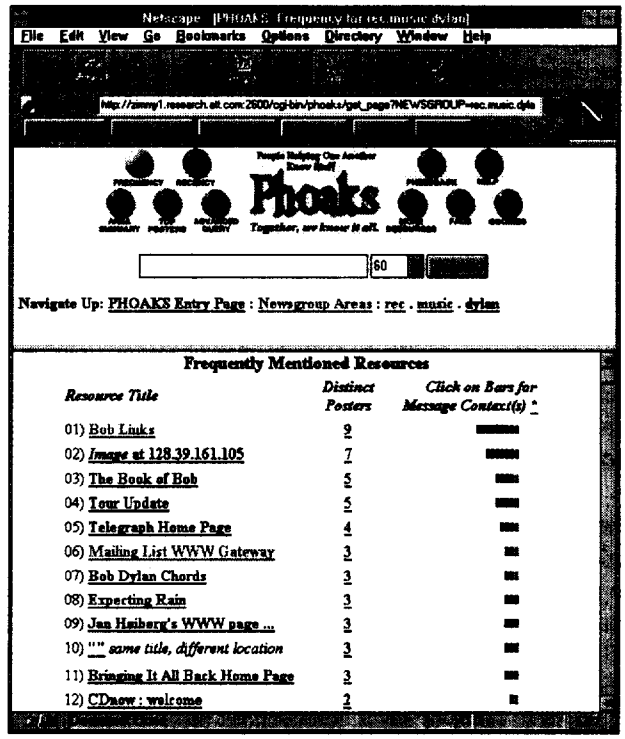

Figure 2: Phoaks Resource Summary Page

We now discuss the interface in a bit more detail, emphasizing how it illustrates the design principles of role specialization, reuse, and (especially) recontextualization. First, Phoaks allows people to play two distinct roles, producers and consumers of recommendations. Second, URL recommendations are reused, i.e., extracted from netnews messages and redistributed. Reuse raises both the opportunity for new connections and the danger of privacy violation; Phoaks has to balance the two. Finally, computing with and presenting contextual information is the very foundation of the Phoaks interface.

Phoaks applies fairly complex rules to the textual context that surrounds URLs to determine which ones are being recommended; further, the interface ranks resources by the number of distinct recommenders. In addition to these computations on context, the Phoaks interface also presents contextual information directly; for example, users can find out who the recommenders for a resource are and other resources they recommended to the newsgroup being viewed. This is useful in at least two ways. First, regular readers of a newsgroup are likely to know who else on the newsgroup has opinions that they trust. Thus, they can investigate just those resources that have been recommended by these people. Second, after investigating and liking a resource, one can find other resources recommended by the people who recommended that resource. Phoaks limits the scope of this feature to work within a single newsgroup. So, if a poster recommends web resources in two different groups, say talk.politics.drugs and comp.lang.java, Phoaks provides no way to correlate the activities. That is, we do not allow name searches, so it is impossible to construct a comprehensive view of a person's posting behavior.

We include hyperlinks to recommender's personal home pages (if we have them) to allow readers to find out more about recommenders. However, we chose not to include "mailto:" links because we thought that by default people who participate in a newsgroup ought not to be easily contacted by people who are not part of the group.

Users also can access opinions about a resource as expressed in the surrounding message context. This is especially important because our classification rules use only limited semantic information. Another important benefit is that a message may mention many resources, some of which did not make it into our summary (e.g., because we could not verify their existence or because we categorized them as home pages), yet users still may visit them when they encounter them in the message context.

Finally, users can see the timeliness of a resource within a community. A "histogram" of shaded boxes is displayed for each resource, with one box for each distinct recommender — the more recent the recommendation, the darker the box. Thus, one can get an impression of whether a resource has been mentioned a lot recently, whether it has been mentioned steadily over time, or whether it appears to have fallen out of favor.

PHOAKS SYSTEM ARCHITECTURE

In developing Phoaks, we faced two hard problems; indeed these are key problems for any conversational data interface. First, algorithms must be developed to recognize the desired information (e.g., recommended resources) accurately and efficiently. Second, interfaces must be created that present the information effectively. Making progress on these problems requires constant iteration of both algorithms and interface, based on user feedback and data analysis.

Processing Conversational Data

Phoaks provides a general architecture for processing conversational data. Our first application of the architecture was a re-implementation of the first URL recommendation prototype; the second application gathers messages that post FAQs. The architecture consists of three main processes:

- *search* — search messages for a specified pattern (such as "http://") and extract contextual information surrounding each instance of the pattern,
- *categorization* — apply rules that classify each instance of the pattern (e.g., URLs used as recommendations vs. personal home pages), and
- *disposition* — process the categorized information (e.g., store it in a database, fetch the content of a URL, assemble separate messages into a single FAQ).

Generality is the first goal of the Phoaks architecture. It is parameterized to allow searching, categorization, and disposition functions for a particular task to be plugged in. *Robustness* is a second goal. Since Phoaks works in a networked environment and processes unstructured and uncontrolled conversational data, it must tolerate and recover from ill-formed data, network errors, and system crashes. *Scale* is another goal. *Portability* is a final goal. We want other people to be able to use Phoaks to create conversational data interfaces; therefore, it does not rely on any commercial software packages or architecture-specific features, so it can be installed and run on any UNIX platform, unencumbered by commercial licenses.

We next describe the three main processes of the architecture in more detail.

Searching

Phoaks begins by seeking syntactic clues that signal interesting data: for example, "http://" signals a URL and "FAQ" or "Frequently Asked Questions" signals a FAQ.

Phoaks can search for literal strings or regular expressions. It can search in either the message header (all of it or only specified lines), the message body, or both. For example, message bodies are searched for URL recommendations, and the subject line is searched for FAQs. Phoaks can search either through an arbitrary specified directory structure (e.g., a netnews hierarchy) or set of files. It keeps track of messages already processed, so it only examines new messages. When searching netnews messages, it is aware of cross-posting, so it processes each message only once. This is significant, since we have found that the average message is cross-posted to 4 different newsgroups.

When Phoaks detects a hit — e.g. "http://" or "FAQ" — it creates a record that stores contextual data from the message. The Phoaks designer can specify which header lines and body lines to include.

Categorization

The next step is to apply rules to categorize a hit; for example, URL mentions must be categorized as recommendations, contact pages, etc. Rules are written in terms of syntactic features of the message, such as the email address of the message poster, line number of the hit, keywords occurring near the hit, etc. (While different conversational data interfaces require different rule sets, many features are useful for a wide variety of tasks.) Thus, features must be extracted from the contextual data before the rules can apply.

Developing a good set of categorization rules is a difficult, intrinsically iterative problem. For example, consider the message signature, a key concept for classifying URLs. How can a signature be recognized automatically? Features like closeness to the end of the message, separators (such as lines containing only "--"), and clue words like the message poster's email address or literals like "homepage", "www", "url", "email" all are helpful. Rule design begins with an examination of a large set of messages. The rules then must be run, data gathered, and their accuracy judged. Real conversational data always hold surprises because there is no formal, agreed upon structure. For example, we might find new ways of separating the message body from the signature. We might find new clue words. And we might discover complications with the "closeness to the end" feature; for example, what if one message replies to another, and includes the replied-to message as a quote at the end? Now the signature is not near the end. Currently, Phoaks rules cover hundreds of conditions leading to 16 categories of URL mentions.

To determine the accuracy of a set of rules, human judges independently classify some data on which the rules have been run, and the human judgments serve as a baseline for judging rule accuracy. There are two aspects of accuracy: *precision* (the percentage of data that the rules classify into a certain category that actually belong to the category) and *recall* (the percentage of data that belong to a category that the rules actually classify into that category. (Our rules for recognizing recommended URLs have 88% precision and 87% recall, for 526 URLs, with inter-rater reliability of 88%.) Phoaks provides several tools to support testing rule accuracy. By default, Phoaks selects a random subset of messages during each run. It writes the messages, along with auxiliary files that contain each hit (e.g., URL), the computed category, and the values of all the features. A separate off-line program presents the hit and surrounding contextual data to a human judge for categorization. Another program collects the judgments of each person who judged the data, computes the inter-rater reliability (the percentage of cases on which the judges agree), and finally computes the precision and recall of the machine's performance, using the data that the human judges agreed on as the benchmark. Phoaks can be run repeatedly on the same set of messages. When a problem is found, this makes it easy to modify the rules and rerun them on the same data to see if the problem has been solved. Of course, true validation of rules requires that they be run on data different from that on which they were developed.

Disposition

After categorizing each hit in a message record, Phoaks performs the specified disposition. The disposition is typically task-specific, although storing information in a database is likely to be done in most cases. (We have built our own database management system rather than using a commercial system, thus helping to keep Phoaks portable.)

For the URL recommendation application, one hard problem is determining when two URLs or two message posters are the same. The same URL may be referred to in textually different ways, such as with or without the default port or a trailing "/", or with inconsistent capitalization. The same person may post from different computers in the same domain or may have several different email addresses, and one person using a single email address may change his or her nickname.

Another problem involves fetching URL content. We fetch the content for three reasons. First, this is a way to verify that the URL is valid (e.g., that it was not a typo). Second, we extract the title to use in our presentation of the URL. Finally, we store a reduced representation (to minimize storage space and processing time and avoid potential copyright problems) of the content, which we use to create indices for our web site and to facilitate user search for relevant newsgroups.

The fetching process can tolerate or recover from failures such as ill-formed HTML, server unavailability, network downtime, and the crash of the machine Phoaks is running on. These are all fairly common occurrences for a program that runs continuously in a networked environment, processing large amounts of real world data.

The disposition function for processing FAQs faces different problems. Often it must assemble a set of messages into a single FAQ ("part 1 of 7", "part 2 of 7", etc.). And it must maintain the most current FAQ, so it must determine when a new FAQ supersedes an old one.

Interface Mechanisms and Techniques

Pages are generated from templates.

We designed a simple page definition language, which can be viewed as an extension to HTML. HTML code is augmented with iteration and conditional constructs and a useful set of special variables. The language makes it very simple to describe, for example, a resource summary page as an iteration over all the recommended resources for a newsgroup (the newsgroup will be supplied when the template is instantiated).

All pages are dynamically generated; frequently accessed pages are pre-generated and cached; all pages are cached after generation.

We designed a cgi script that responds to requests for Phoaks pages. A page request is a query URL. We have a well-defined query language of attributes (such as page-type and newsgroup) and legal values (such as resource-summary).

When a page is requested, the script checks to see if the page has been cached. If so, it returns the page. If not, the page is generated, cached, and then returned. By analyzing access to the Phoaks web, we found that about 75% of the pages accessed are resource summary or index pages. Therefore, we run queries to generate and pre-cache these pages fresh every time we complete a run through netnews and update our databases. Thus, the most accessed pages will already exist whenever they are requested.

The query language also supports programmatic access: a program can construct the proper query URL to pass to our server. Given the proper query attribute, Phoaks returns the results as simple, structured text (rather than HTML). The intent is to allow others to experiment with creating their own interfaces to Phoaks data.

We "spoon feed" spiders (indexing programs for search engines) specially designed content to facilitate indexing of our pages.

In our original design, resource summary pages included message context, excerpts from up to 50 messages that mentioned the URLs contained on the page. We did this to gain the significant advantage of popular search engine spiders indexing our pages richly. Currently some Alta Vista queries contain Phoaks pages as the first and/or second returned page, for example: "Stephen King", "Rush Limbaugh", "country western music", "pc hardware chips". By analyzing access data, we found that bursts of spider access were followed by bursts of access by people. This pattern was true both for access to our site in general and for specific newsgroups.

However, including message context had a significant price: it made our pages large, thus increasing download time which matters to human users. A goal of our current design was to decrease page size for humans but still retain the indexing advantage via spiders.

To decrease page size, we created separate message context pages. Keeping our pages highly indexed required more work. First, we modified the cgi script that handles requests for Phoaks pages to check if the requesting host is a known spider (e.g., Alta Vista's spider comes from scooter.pa-x.dec.com with a particular user-agent name). Then, if a resource summary page is requested, we return special indexing text created by concatenating the reduced representation of the content of all the pages that the resource summary page links to. Since Phoaks pages serve as link pages — people visit them to find content that they actually are interested in — this is an accurate representation of the content users can access via any particular Phoaks page. One other aspect of the cgi script's response to spiders is that it lets us control which of our pages do and do not get indexed. Specifically, we do not let the pages for individual posters get indexed. This solves a problem with our earlier design — some people complained to us because a Phoaks page was indexed more highly than their personal home page (for queries they felt should find them), and other people did not want to show up in our pages at all. Furthermore, search engines could be used to correlate poster activity across newsgroups via information provided on Phoaks. We had inadvertently violated one of our design principles, respecting the privacy concerns of the producers of conversational data. (Note: Phoaks respects the **x-no-archive: yes** protocol, so posters may keep their data private.)

We provide feedback mechanisms that attempt to leave ownership of the conversational data with the people who produced and use it.

The three most common types of feedback we receive from users are requests to add a link to one of our pages and notifications that the URL of a resource has changed or that it is off-topic for the group theme. We have implemented forms that lets users add links, judge resources as being good or off-topic, and submit URL change notices; a few users give us each type of feedback daily. In effect, we treat the automatically constructed pages as a rough draft that we expect human experts to help refine. This is a kind of participatory design [17] of content. One of the interesting interface questions is just how to communicate this "rough draft" status of Phoaks information and further how to communicate "polished

draft" status to pages once a significant number of readers have corrected any machine classification errors.

USING PHOAKS TO BUILD A NEW CONVERSATIONAL DATA INTERFACE

To summarize the scope of the Phoaks architecture, we briefly describe the steps involved in creating a new conversational data interface system.

- Stream selection — specifying the source of the conversational data to be processed; for example, its location in a filesystem.
- regular expression definition — stating the regular expression that defines the information to be extracted from messages and processed.
- message feature definition — determining additional features needed for categorizing data if the standard set of features supplied with Phoaks is inadequate.
- category definition — the regular expression defines the set of data that will be extracted; the designer then must specify the different categories into which this data should be categorized.
- rule definition and organization — stating rules to recognize each category; ordering the rules appropriately (for example, the Phoaks rules for recognizing URLs in signatures come before the recommendation rules; this means that a recommendation is something that is not in a signature).
- disposition definition — defining what is to be done with categorized data, e.g., the format for storing data.
- rule validation — categorizing sample data that Phoaks has set aside, then running the program that computes inter-rater reliability, precision, and recall, iterating as necessary.
- template definition — specifying how the conversational data that has been extracted, stored, and categorized should be presented to users, including links between different views of the data.

DISCUSSION: GENERAL LESSONS FOR CONVERSATIONAL DATA INTERFACES

Feedback from users has taught us several important lessons about creating interfaces to conversational data. In general, visitors find the Phoaks web site useful. For example, two hundred other sites now point at Phoaks.

First, a few people expressed concern that making certain information explicit could harm the social fabric of a community. For example, in our original design we used language like the "top 10" resources and the "top contributors" to a newsgroup. Perhaps this could cause people to compete for positions on a list, thus posting lots of URLs or getting all their friends to post their URLs. We responded to this concern by modifying the language used in the interface to try to minimize the impression of competition among people and URLs.

Second, we confirmed the danger of labeling or describing people's activity. In particular, in our original design we described people as "recommenders" and we referred to "recommended resources". We used these terms, of course, to refer to resources that our rules had classified as recommendations. However, even though our rules have good precision, certain rare false positive errors have very high cost. For example, a person posted a message condemning the contents of a URL operated by a hate group. When this person found the relevant Phoaks page and saw that he was described as a "recommender" of the URL, he was understandably upset. We responded to this problem by replacing terms like "recommend" with more neutral terms like "post". In the newest version of the system, we also have added extensive help text that explains our terms and makes it clear that posting a resource does not always count as an endorsement of that resource. We also point out that going to the message context for a resource will give some indication of what the poster was saying about it.

This raises a general issue in the design of interfaces to conversational data: does a system like Phoaks offer a faithful history of a conversation, or does it create a more abstract memory? For example, do the people who participate in soc.culture.african-american want the URL of a white supremacist group to show up in Phoaks when it shows up in netnews? Does it matter whether it was posted by a member of the community who was condemning it, or by an outsider who was attacking the community by posting it? These are the type of thorny issues that any conversational data interface face, and that can only be addressed through dialogue with the producers and users of the conversational data.

To conclude, Phoaks is an experimental system for building collaborative filtering interfaces to conversational data. The results of the experiment so far are encouraging. Usenet news messages are a rich source of URL recommendations, and the recommendations can be extracted fairly accurately and efficiently and ranked automatically. Our web site summarizes recommendations and associated contextual information for about 1500 newsgroups, and, as of December 1996, attracted about 3,500 users per day and generally positive user feedback.

REFERENCES

1. Ackerman, M. and McDonald, D., Answer Garden 2: Merging Organizational Memory with Collaborative Help. in Proceedings of CSCW'96 (Boston MA, November 1996), ACM Press, 97-105.

2. Allen, R.B. User models: Theory, method and practice. International Journal of Man-Machine Studies, 32 (1990), 511-543.

3. Brewer, R.S. and Johnson, P.M., Collaborative Classification and Evaluation of Usenet, University of Hawaii Technical Report ICS-TR-93-22, October, 1994.

4. Brothers, L., Hollan, J., Nielsen, J., Stornetta, S., Abney, S., Furnas, G., and Littman, M., Supporting informal communication via ephemeral interest groups. in Proceedings of CSCW'92 (Toronto Ontario, Canada, November 1992), 84-90.

5. Cohen, W.W. Learning to Classify English Text with ILP Methods, In: Advances in Inductive Logic Programming (Ed. L. De Raedt), IOS Press, 1995, to appear in IOS Frontiers in AI and Applications series.

6. Fischer, G. and Stevens, C. Information Access in Complex, Poorly Structured Information Spaces. in Proceedings of CHI'91 (New Orleans LA, April 1991), ACM Press, 63-70.

7. Goldberg, D., Nichols, D., Oki, B.M. and Terry, D. Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM, 35, 12 (December 1992), 51-60.

8. Grudin, J., Social Evaluation of the User Interface: Who Does the Work and Who Gets the BENEFIT? in Proceedings of INTERACT'87 (Amsterdam, The Netherlands), 805-811.

9. Hammond, K.J. Burke, R., Martin, C., and Lytinen, S. FAQ Finder: A Case-Based Approach to Knowledge Navigation. In AAAI Symposium on Information Gathering in Heterogeneous, Distributed Environments. (Stanford University, March 1995), AAAI Press.

10. Hill, W. C., Hollan, J. D., Wroblewski, D., and McCandless, T. Edit Wear and Read Wear. in Proceedings of CHI'92 (Monterey CA, May 1992), ACM Press, 3-9.

11. Hill, W.C., Hollan, J.D. (1994) History-Enriched Digital Objects: Prototypes and Policy Issues. The Information Society, 10, 139-145.

12. Hill, W.C., Stead, L., Rosenstein, M. and Furnas, G. Recommending and Evaluating Choices in a Virtual Community of Use. in Proceedings of CHI'95 (Denver CO, May 1995), ACM Press, 194-201.

13. Hill, W. C. and Terveen, L. G. Using Frequency-of-Mention in Public Conversations for Social Filtering. in Proceedings of CSCW'96 (Boston MA, November 1996), ACM Press, 106-112.

14. Malone, T.W., Grant, K.R., Turbak, F.A., Brobst, S.A. and Cohen, M.D. Intelligent Information Sharing Systems. Communications of the ACM, 30, 5 (May 1995), 390-402.

15. Maltz, D., and Ehrlich, K. Pointing the Way: Active Collaborative Filtering. in Proceedings of CHI'95 (Denver CO, May 1995), ACM Press, 202-209.

16. Marx, M. and Schmandt, C. CLUES: Dynamic Personalized Message Filtering. in Proceedings of CSCW'96 (Boston MA, November 1996), ACM Press, 113-121.

17. Muller, M. J. and Kuhn S. (eds.) Communications of the ACM, Special Issue on Participatory Design, 36, 6 (June 1993).

18. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. in Proceedings of CSCW'94 (Chapel Hill NC, October 1994), ACM Press, 175-186.

19. Shardanand, U., and Maes, P. Social Information Filtering: Algorithms for Automating "Word of Mouth". in Proceedings of CHI'95 (Denver CO, May 1995), ACM Press, 210-217.

20. Stevens, C. Automating the creation of information filters. Communications of the ACM, 35, 12 (December 1992), 48.

21. Terveen, L.G., Hill, W.C., Amento, B., McDonald, D., and Creter, J. Phoaks: A System for Recognizing and Sharing Recommendations. To appear in Communications of the ACM, special issue on recommender systems, April 1997.

22. Whittaker, S. Electronic Collaboration: An Empirical Evaluation of Factors Affecting Mediated Group Interaction. in Proceedings of CSCW'96 (Boston MA, November 1996), ACM Press, 409-418.

23. Wittenburg, K., Das, D., Hill, W., Stead, L. Group Asynchronous Browsing on the World Wide Web. in Proceedings of Fourth International World Wide Web Conference (Boston MA, December 1995), O'Reilly & Associates, 51-62.

24. Wroblewski, D., McCandless, T., Hill, W. (1995) Advertisements, Proxies and Wear: Three Methods for Feedback in Interactive Systems, in Dialogue and Instruction, Beun, R., Baker, M., and Reiner, M. editors. Springer-Verlag, 336-347.

25. Yan, T. SIFT: A Tool for Wide-Area Information Dissemination. in Proceedings of USENIX '95 (New Orleans LA, January 1995) , 177-186.