Beehive: A System for Cooperative Filtering and Sharing of Information

Bernardo A. Huberman and Michael Kaminsky

Dynamics of Computation Group
Xerox Palo Alto Research Center
Palo Alto, CA 94304

Abstract

We have designed and implemented a distributed system for social sharing and filtering of information. It relies on the automatic recording of the behavior and interactions of members of communities of practice. The system automatically updates membership in informal communities at regular intervals and provides a simple and intuitive interface for distributing relevant information among its members.

August 19, 1996

1. Introduction

An important consequence of the existence of informal communities of people is the efficiency with which members of these groups share and filter relevant information. In spite of the fact that most people are bombarded with very large number of news items on a daily basis, recommendations from colleagues and friends can sharply reduce the amount of noise that pervades daily life. It is therefore natural that in light of the information explosion created by the Internet, several efforts have been reported to create information tools that mimic those processes of recommendation [1].

Among such attempts are systems like *GroupLens* [5], *Firefly* [6], and the *Bellcore Video Recommendation Guide* [3]. While very useful, they rely on a universal knowledge of all the interactions among all users, musical rankings in the case of *Firefly*. This implies that a single server holds the full membership of the user group, thus creating access problems when many users attempt to use the system. Finally, there is an implicit critical mass of users that is needed before these systems become useful as an information sharing tool.

In order to improve on those shortcomings, we have designed and implemented a distributed system, *Beehive*, for social information and filtering of information through the behavior and interactions of members of communities of practice [4]. *Beehive* automatically updates membership in informal communities at regular intervals and provides a simple and intuitive interface for distributing relevant information among its members.

The *Beehive* system works in the following way. First, a computer program that we have implemented creates a table of who interacts with a given individual, and how strongly, based on the frequency with which these community members communicate with each other. Over a period of days, weeks, or months, the data in the table reflects all members of the communities with whom the individual is exchanging messages. The user then sets a frequency threshold above which automatically defines active membership in a particular community.

Such a table and the information residing therein is used in such a way that every time a member of a group finds something of interest, a copy of the pointer

(in the form of a URL or an email message) can be made available to all members of a given community. This is done through the simple act of selecting the item and clicking the icon symbolizing the community.

The result of these simple operations is that those people frequently interacting with each other will have access to the information collected and filtered by them. The periodic updating of the community interaction table means that there is no need to remove oneself from old mailing lists or add oneself to new ones.

2. System Description

The *Beehive* system is divided into two distinct parts, each implemented as a separate program. The first one is the information gathering segment of the system, and it consists of a perl script that generates data for the user-interface segment of the *Beehive* system. It does so by searching headers of the user's mail spool and constructing a directory of members of that community.

The second part of the *Beehive* system is the program itself, which, written in C with the X and Motif libraries, is a user interface to the directory generated by the perl script. This interface allows the user to select a mail message, a URL, or any other text on the screen, and to send it to the members of one of the user-specified communities by simply clicking on its representative icon.

The way the information gathering segment of the system works is by searching through standard RFC822-style mail headers for either "To:" or "From:" lines—based on a command-line option—and tabulating the total number of messages sent to or received from each found email address. From these totals, the perl script generates an interaction index by dividing the number of email messages containing a given address (in one of the two specified headers) by the total number of addresses encountered while parsing the mailspool (again in the specified header).

By default, the perl script outputs a directory readable by the *Beehive* user-interface segment of the system. An optional argument, however, directs the perl script to additionally output a table of addresses with their respective occurrence

frequencies and interaction indices. This supplemental information gives the user more input to aid him or her in constructing a communities description file.

Once the perl script generates an internal table of interaction indices, it checks to see if the email addresses for a given community have interaction indices that are greater than or equal to the community's threshold. This threshold is determined on a per-community basis by the user. Those addresses satisfying this condition become part of the generated communities directory.

The communities description file is a user-created definition of his or her personal communities. The format of the file is one community per line as follows: **community name, threshold, email addresses**. The community name may contain spaces, but no commas, the threshold is a real number, and the email addresses are space separated.

The second component of the *Beehive* system is the graphical user interface, which, when it starts up reads the communities directory into memory and then opens a window on the user's display. For each community in the data file, the program generates a corresponding button with an icon; additionally, a quit button (with no icon) is placed above the others.

To use the *Beehive* program, a user selects text with the mouse that is to be sent to other members of a given community. While the text is selected, the user clicks the button corresponding to the community to which the text is to be emailed. The *Beehive* program grabs the selected text from the X cut buffer, tacks on a subject line, and distributes the mail to all of the interacting members of the chosen community.

The *Beehive* program always uses the communities directory in memory which is loaded only once when the program is run. When the perl script is re-executed to generate a new communities directory, the user must quit *Beehive* and run it again.

We envision the *Beehive* program as a persistent window on the user's desktop which is accessible at any time. Users can, in a virtually transparent way,

disseminate important and fascinating information seamlessly to others who have similar interests.

3. An Example

The first step in the *Beehive* process is to create a preliminary set of communities from which the perl script can create a communities directory. The communities are described by the user in a text file such as the one below:

```
friends, 0.01, address1@somewhere.edu
researchers, 0.03, address2@hello.edu address3@elsewhere.com
pizza-lovers, 0.1, yummy@pizza.place
nobodygroup, 0.2,
```

Each line defines a different community with a name, threshold, and list of email addresses of potential members. A user sets up the desired communities before running the perl script. Typically, these communities will remain static between instances of running the script but may be added or subtracted from at any time.

Step two, running the perl script, analyzes the user's mail spool and generates an internal table of interaction indices. In this case, for the "friends" community, the script will determine such an index for the address "address1@somewhere.com." If that number is higher than 0.01, the community's threshold, "address1@somewhere.com" will be considered part of the "friends" community in the final communities directory. If not, this person is not currently considered by the user to be an interacting member of his "friends" community and will not receive email that the user sends to that community. The script repeats the same process for each community and email address in the initial communities description file.

An optional argument to the script directs the script to output a table indicating the frequency and interaction index for each found email address. A sample excerpt from such a list is as follows:

```
                                Number of       Interaction
Address(es)                     Occurrences     Index
-------------------------------------------------------------
address1@somewhere.edu            3             0.012821
address2@hello.edu                1             0.004274
william@the.net                   1             0.004274
joel@global.village               7             0.029915
address3@elsewhere.com            17            0.072650
surya@quantum.physics.gov         3             0.012821
film@classics.movies.org          4             0.017094
help@at.last                      8             0.034188
food@all.night.long               2             0.008547
novels@reallyneatbooks.com        5             0.021368
applesauce@goodfood.com           4             0.017094
tomatoes@badfood.com              7             0.029915
guitar@fun.instruments            8             0.034188
```

Step three consists in redirecting the output of the perl script—the communities directory—to a file to be used by the *Beehive* program. The directory generated from the above communities description file looks as follows:

```
friends
 address1@somewhere.edu
researchers
 address3@elsewhere.com
pizza-lovers

nobodygroup
```

This file is given as a command line argument to the *Beehive* program which in this example will create a window on the user's X display similar to the following:

Each icon is a button and represents a community in the communities directory. A user begins by first selecting text in any window on his or her display. Then, the user simply clicks the button of the community to which the text is to be emailed and the *Beehive* system does so transparently. The *Beehive* program can run in the background and sit inconspicuously on the user's desktop, immediately available when he or she sees an interesting or important piece of information. As the user communicates over time with current community members and other non-community members, the perl script is re-executed to reflect these changes.

4. Discussion

We have described a new system for providing automated sharing and filtering of information among members of informal communities, without the need to build a centralized database. Based on personal preferences, the *Beehive* system automatically determines the varying membership in the communities that are relevant to each member and provides a way of filtering and forwarding information to each other.

The advantage of this scheme over other methods is that the finding of those individuals related by interest is done automatically and updated on a frequent basis. Future enhancements may include a continuously updating interaction table, providing dynamically varying communities of practice. Although it is convenient to think of the *Beehive* system as a World Wide Web tool, it can also be useful in the context of net-news or email.

An important difference between the *Beehive* system and other collaborative filtering systems lies in the distributed nature of the former. By avoiding a centralized database, *Beehive* meshes naturally in the current computing paradigm which is globally distributed. Another advantage of this system is that unlike implementations such as *Firefly* [6], *URN* [2], and *Bellcore's Video Recommendation Guide* [3] it does not restrict the set of filterable information to a specific subject or distribution medium whether it be music, movies, or Internet newsgroups (USENET). Such restrictions not only mandate server-side recommendation databases but a very specific user interface. *Firefly* must be accessed via the World Wide Web and *URN* through a single modified news reader.

Beehive's implementation by contrast, is general enough so as to be useful in any context regardless of the content of the medium. Information on movies and music, email and news articles alike can be easily distributed to communities. The act of information sharing itself which is achieved by just using the *Beehive* system, replaces the filtering, weighting, and voting processes that need to be made explicit in other collaborative filtering systems [2].

The *Beehive* system is not without shortcomings as it currently requires a nominal amount of user intervention to decide on the timing of updates for the communities directory. Since every user has different behaviors with respect to email and web browsing, the choice of the timing parameter depends on per-user factors such as how much email sent and received and the number of different people with which the user communicates.

Nevertheless, the *Beehive* system provides a generalized interface to a fully distributed collaborative filtering and information sharing mechanism. This is particularly relevant at a time when the growth Internet and the associated explosion of information will make informal communities increasingly important vehicles for the discovery of relevant information.

References

[1] Workshop on collaborative filtering. School of Information Management and Systems, University of California, Berkeley, March 1996.

[2] Robert S. Brewer and Philip M. Johnson. Collaborative classification and evaluation of usenet. *University of Hawaii, Department of Information and Computer Sciences*, (CSDL-TR-93-13), 1994.

[3] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. *http://community.bellcore.com/navigation/videos.html*, 1994.

[4] Bernardo A. Huberman and T. Hogg. Communities of practice: Performance and evolution. *Computational and Mathematical Organization Theory*, 1:73–92, 1995.

[5] Paul Resnick, Neophytos Iacovou, Mitesh Sushak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the CSCW 1994 Conference*, 1994.

[6] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". *CHI '95*, pages 210–217, 1995.