URN: A New Way To Think About Usenet

Robert Brewer

Collaborative Software Development Laboratory
Department of Information and Computer Sciences
University of Hawaii
Honolulu, HI 96822
(808) 956-3489
rbrewer@uhics.ics.hawaii.edu

CSDL-TR-93-06

Last Revised: May 1, 1993

Support for this research was provided in part by the National Science Foundation Research Initiation Award CCR-9110861 and the University of Hawaii Research Council Seed Money Awards R-91-867-F-728-B-270 and F-92-868-F-728-B-430.

Contents

1 Introduction

This document presents an overview of the URN project. It is intended to show the motivation behind the project as well as describing the current status of the research.

The URN project is an Egret application that implements a new paradigm for utilizing the massive, ever-changing data stream that is Usenet (for background information about Usenet, see Section 2). The acronym URN stands for Ultimate Read News, but it can also be thought of more symbolically as a repository for knowledge. Egret is an environment for the implementation of exploratory collaborative hypertext applications which is described elsewhere [Johnson, 1991; Johnson, 1992]. The goal of URN is to make the information available through Usenet more useful by using different perspectives on Usenet and new ways to extract useful data from Usenet. We claim that these ideas constitute a paradigm shift in the way we view Usenet. This new paradigm has three components: agents that scan Usenet for useful information on a user's behalf, knowledge condensation techniques that allow Usenet to be reorganized and annotated, and knowledge communities that provide explicit representation of authors and readers. This new paradigm grew out of solutions to the three major problems with Usenet: there is too much information, it is too ephemeral, and its current representations are too limited.

The next section will briefly describe Usenet to those unfamiliar with it. Discussion of the three problems with Usenet and our three proposed solutions will follow. The next section lays out the research methodology and the evaluation process. We then describe the current status of the research methodology and the research tool implementation. The final section discusses the future direction of the project.

2 Background

Usenet (standing for Users Net) is a massive but loosely connected network of computers that exchange 'netnews' which can be thought of as a kind of 'public' email. Any user on a Usenet node can post an article to Usenet by simply typing in some text and submitting it to a program on the local computer. This local computer then forwards the article to a few close-by Usenet nodes, who in turn forward it in turn to other nodes. In this manner news is propagated around the world, yet the original posting machine need only send it to a few near-by machines. Although Usenet started with only a few nodes, its growth has been incredible.

As of March 1993, an estimated 76,000 Usenet sites exist with over 2.4 million

Usenet readers[1]. The traffic on Usenet is enormous: in the week of April 5, 1993, for example, Usenet generated approximately 400 megabytes of data, consisting of approximately 176,000 separate articles. This is typical of Usenet traffic.

Usenet articles are categorized into hundreds of 'newsgroups'. These newsgroups are the primary way in which articles are broken down in to different subject areas. Newsgroups are hierarchically named with the levels of the hierarchy separated by '.'. For example, the newsgroup about Macintosh hardware is called 'comp.sys.mac.hardware' and the newsgroup for IBM compatible hardware is called 'comp.sys.ibm.pc.hardware' while the newsgroup about Star Trek information is called 'rec.arts.startrek.info'. The content of these newsgroups is highly varied, ranging from groups about software engineering ('comp.software-eng') to groups about dogs ('rec.pets.dogs') to groups about abortion ('talk.abortion').

Just as in all communications media, articles can be statements, questions, comments, replies to questions, poems, or any other textual object. After an article is posted, other users may choose to 'followup' that article in such a way that the followup article is linked to the original article. This process is recursive, as other users can create followups to followups. A set of articles linked together in this way on a common topic is called a 'newsthread' or simply thread. Sophisticated newsreading software (such as trn or GNUS) is capable of recognizing these threads and allows the user to explore Usenet at this level. Follow up articles often include quotes from the original article. This quoting is usually done in an automated fashion so that readers can distinguish between quoted and original text. However, this makes for a great deal of repeated information, especially in long threads. In fact in March 1993, quotes appeared as more than 9% of Usenet's volume. As an aside, not all articles posted to Usenet are human-readable text; some are encoded versions of binary files: applications, pictures, and sounds.

3 Motivation

3.1 Three Problems with Usenet

3.1.1 Too Much Information

With some understanding of what Usenet is, we can now describe some of the problems of Usenet that prevent it from being utilized to its full potential. From the description

[1]Data from Brian Reid's <reid@decwrl.dec.com> postings in the news.lists newsgroup with message-ID <1ptj0d$2h1@usenet.pa.dec.com>.

above the first problem with Usenet is clear: too much information. With as much as 1.8 **gigabytes** of new articles each month, it is impossible for any human to actually read a sizable fraction of Usenet, let alone all of it. In one sense, this isn't important because presumably no one is interested in every issue or topic. However, the primary way that people filter out information is through subscription to newsgroups. Subscribed newsgroups are those that one's newsreader checks when looking for unread articles to display[2]. Newsgroups that one is not subscribed to are therefore out of one's way, so by carefully choosing which newsgroups to subscribe to one can control the volume of information one reads.

Unfortunately, filtering via newsgroup subscription is not a solution to the problem for many users. If these newsgroups have high traffic then one is confronted with possibly hundreds of new articles each day, too much for the average person to read. The problem is further complicated by the expiration of articles: unlike most printed material, Usenet articles are expired on a regular basis. If one does not read news regularly, articles are deleted before one even sees them (see Section 3.1.2). The high traffic in newsgroups is often caused by the ease of posting an article. For many end users there is no cost to post an article to Usenet and its 2.4 million readers. Since they perceive no cost, they sometimes post recklessly, apparently without thinking. These reckless posts are often: poorly written, uncourteous, repeated information, and incorrect information. Due to these reasons there is a lot of 'noise' on Usenet. The information explosion will only get worse as more sites and more people are connected to Usenet at lower and lower cost.

3.1.2 Too Ephemeral

Another problem with Usenet is its lack of persistence. As previously described, the amount of information available in Usenet is staggering and continually changing. Every day each Usenet node receives megabytes of new articles. Since disk space is finite, eventually old articles must be removed from disk to accommodate new articles. It is almost universally the case that old articles are removed by deletion. Since there is very little in the way of archival of Usenet, once old articles are 'expired' they go away forever, unless some user of Usenet decided to save that particular article for future reference. While it is possible to archive Usenet (in fact there is a company that sells

[2]Subscription is therefore somewhat of a misnomer as each user has their own list of subscribed newsgroups, but their Usenet node only has one copy of the articles. In other words, a newsgroup subscription is completely unlike a magazine subscription where each subscriber receives a separate copy of each issue.

subscriptions [in the magazine sense] to Usenet via CD-ROM[3]), such an archive would not be very useful as a reference tool because any search query would return many false matches due to the massive amount of noise present in Usenet.

Usenet also lacks the kind of condensation that occurs in other media. For example, if one wants to know what happened yesterday one might read a daily newspaper. If one wants to know what happened last year, one might read a magazine or journal. If one wants to know what happened 50 years ago one would probably refer to a history book. In this fashion information is condensed from the voluminous and noisy level of a newspaper to the concise version found in history books. Condensation is made difficult in Usenet because of the expiration of old news. Of course the readers of Usenet may remember what has occurred and might be able to respond to a question concerning old information, but this is an unreliable system at best. One exception to the lack of condensation in Usenet are FAQ (Frequently Asked Question) files. These files are created by hand in an *ad hoc* fashion with the goal of answering all of the most commonly asked questions in a newsgroup. Then the FAQ file is frequently posted to the group in the hope that new subscribers will read the file and find an answer to any basic question they might have instead of posting the question to the group. While this facility is quite useful and does reduce noise, it is completely driven by the philanthropy of the FAQ creator and the type of questions addressed are usually basic and factual.

3.1.3 Representation Too Limited

The final problem of Usenet that we will discuss is the document representation used by all current newsreading software. At its most basic level, Usenet consists of a stream of different articles. Originally these articles were only broken up by which newsgroups they were posted to. In recent times, articles contain information that indicates which other articles it is referring to. With this information we can build tree structures where the first article is the root of the tree and the followups are connected to the root by various numbers of links. These trees are called threads. This thread is a very valuable entity because it gives us a level of abstraction between the article (too specific) and the newsgroup (too vague). Because threads consist of multiple articles, it follows that there are fewer threads than there are articles. So if we can just find the threads we are interested in, we can get at the information we want to see.

Unfortunately, threads have a big problem. Often one thread actually contains

[3]Sterling Software, email: `cdnews@Sterling.COM`, information files: `ftp.uu.net: /vendor /sterling`

multiple different and only vaguely related topics. Since the news posting programs have no natural language understanding capabilities, they cannot tell whether an article to be posted is on the same topic or not. Current newsreaders also restrict threads to a single group so a thread that jumps into another group will seem to disappear from the original group even though the thread has merely shifted to another group. This is typical of the current Usenet paradigm in that newsgroups tend to be rigidly separated despite different newsgroups being quite often related.

The current Usenet paradigm is a hierarchy of articles; threads are groups of articles and newsgroups are groups of threads. This view fails to capture important information, namely the representation of authors.

3.2 Three Solutions

3.2.1 Agents Search Usenet on Users' Behalf

The solution to the problem of information explosion (for the people reading Usenet at least) is to figure out what kinds of things the user wants to see and display them (a 'Do What I Mean' interface). It should be the goal of any newsreader to show the user only articles that the user is interested in. Current newsreaders already attempt to provide this with the use of 'kill files'. A kill file is a file that contains a list of patterns and associated actions. Any article that matches one of the patterns in the file has the associated action applied to it. While in principle it is possible to use these files to perform any action, typically the action is to mark the article as read ('kill' it). This allows a user to exclude from view articles on a certain subject, or articles from a certain person, etc. While this facility is quite powerful, it is sorely underutilized by readers of Usenet. The primary reason for the lack of use is that it takes a substantial amount of effort, intellectual and otherwise, to come up with a list of patterns for articles that one doesn't like.

There is another problem with kill files: they can only exclude articles, they cannot bring articles to the users attention. Since Usenet is very large, it is much more logical to think about patterns that match things that one does like than to come up with patterns that exclude everything that one doesn't like. In addition, kill files also run the risk of killing off articles that the user does in fact want to read. For example, a hypothetical user might dislike IBM, and therefore create a kill file entry that kills all articles related to the subject of IBM. However, this hypothetical user might also love Apple Computer, and really like to see articles about Apple. In this case, if an article was posted that was about a partnership between Apple and IBM the kill file would

kill it, even though it is likely that the user would want to see the article.

Given the disadvantages of kill files we have decided to use weighting functions. A weighting function takes an article as input, and returns an integer value representing the 'weight' it assigns to that article. The weighting function consists of a field to be searched, a regular expression, and a weight to be assigned to articles matching the regular expression. The weight can be positive or negative depending on what the goal of the particular function is. Returning to our hypothetical Apple lover, he or she might have a weighting function that assigns a moderately negative weight to articles related to IBM, and a weighting function that assigns a high positive value to articles related to Apple.

```
Field:  Subject, Reg Expression:  ".*[Aa]pple.*", Weight:  +50
Field:  Subject, Reg Expression:  ".*IBM.*", Weight:  -35
```

All the weighting functions are run over all the available articles, and the weights for each article are added up. Articles are then sorted in decreasing order by weight, hopefully leaving 'good' articles at the top and 'bad' articles at the bottom[4]. Given the weights described above, the Apple/IBM partnership article would get a low positive rating from our Apple lover which is presumably what he or she desires. By assigning the absolute value of the weight of the IBM function to be less than the weight of the Apple function we are using the rule "innocent until proven guilty", i.e. the Apple lover would rather see an article about IBM than miss seeing one about Apple.

While such a weighting function overcomes one disadvantage of kill files, it shares the other: the intellectual effort required to create them. We will solve this second problem by monitoring user actions and automatically generating a weighting function from them. The metrics we will employ are: how long was spent reading the article, was the article read completely, and was the article archived (see Section 3.2.2 for more information on archival). When the system thinks it has inferred a useful weighting function it will present its finding to the user and ask the user if it should be used in the future or how it should be modified. With the powerful tool of automated weighting functions it should be possible to scan much larger portions of Usenet than the user would otherwise read.

There is also no reason for the weighting process to be interactive. It can be done in the dead of night when computer resources are more available and presented to the user the next time he or she uses URN. By searching all of Usenet with these weighting functions users can get more information than they would normally be able to (because

[4]Perhaps 'weight' is somewhat of a misnomer considering that articles with high weight rise to the top. Such is life.

of the high volume) and spend less time reading noise.

3.2.2 Knowledge Condensation

The process of knowledge condensation requires two elements: persistent archival, and information reorganization. The solution to the lack of persistence is sophisticated archival support. When people read Usenet, they often find articles that they would like to save for future reference. Unfortunately, newsreaders don't supply any way to do this in a reasonable fashion. Some people save the articles into one large file per newsgroup, some save each file with a different (and hopefully descriptive) filename, and some people forward the article to themselves via email and then file it as they would email message. The main problem with all of these methods is that there is no good way to refer to this information at some future date. These archived messages will represent a more concise version of Usenet because people presumably don't archive pure noise, so some signal is archived. URN will provide a built-in mechanism for personal Usenet archives thus standardizing the archives, and we will also allow queries to this personal archive making it much more useful than a directory full of files. Since the archives will be in a uniform format, it will be possible to combine all the personal archives at one site into one super archive that contains all the articles that users felt were worth saving. In this way Usenet can be condensed into a much smaller and lower noise format making it a more useful reference source.

Information reorganization will be facilitated by the dissection of articles into their semantic components. These components can then be reorganized by changing which components are linked together and users will be able to annotate components. For example, threads often consist of an initial post which is then quoted and responded to point by point in reply articles. This provides a natural division of articles into components: each point in the original article is a unit and each response to each original point is a unit. Users of URN may then choose to make a new series of links through these sub-article units, creating a completely new way of viewing the same information which is more concise than the original thread of articles. Together persistent archival and information reorganization will allow the condensation of Usenet into an archive of useful knowledge.

3.2.3 Explicit Author Perspective

Viewing articles as threads and newsgroups is useful but these views are only simple aggregations of articles. One view that is not used but follows easily from the data of

Usenet is the author view. By collecting data on who posts where and what they post, we can get an idea about what a person's interests are, allowing us to see Usenet in a completely different way: as communities of users. When reading a paper document we often want to know the background of the author. With the author view we are able to see the background of an author when reading her or his article. By creating a list of authors who appear to be experts on a subject, users can send email to these experts asking for help or advice instead of posting an article to a newsgroup which might be seen by tens of thousands of users.

Knowledge about an author can also be collected from a more delicate source: the author's Usenet archive. The set of articles a person chooses to archive is probably even more representative of their interests than the articles they post (primarily because archiving is easier than posting, so people archive more often than they post). However it is important to insure that users' privacy is not violated by describing what they choose to archive, which might be quite personal. One way to avoid any breach of privacy would be for the system to attempt to infer from a user's archive what their interests are and then present the user with the system's hypothesis and allow the user to edit it before it is broadcast to others. While the 'public' persona of an author can be reverse engineered from the postings they make, the 'private' persona can only be determined by the newsreader of the author. This means that the private persona must be transmitted to other users if they are to make use of it. One simple way is to only share personae among other users in a particular geographical location or site (for example only sharing personae within the University of Hawaii or only within the Collaborative Software Development Group). Another more complex alternative is to set up a centralized site that collects and disseminates private personae.

4 Approach

The URN project will proceed in the following way: the system will be implemented, the system will be used to generate data on its effectiveness, and this data will be evaluated to determine if the system achieved its goals.

The implementation of the system will satisfy the requirements mentioned above in the solutions to the 3 problems of Usenet. URN is implemented on top of Egret which is in turn based on Lucid Emacs and a HyperBase server. Egret is an excellent system for implementing URN because of its support for hypertextual information and tight integration with Lucid Emacs. While Egret does provide a flexible research environment, it does so at the expense of speedy execution. Since URN will be initially

unable to provide the speed and volume of a conventional unsophisticated newsreader it will not attempt to provide all the low-level services such newsreaders supply (such as automatic binary extraction or macros). Instead, it will provide only the essentials of reading and responding to articles and following threads (as a framework for the more advanced ideas described earlier).

4.1 Performance Evaluation

The proper evaluation of a system like URN is somewhat difficult in that its goals are exploratory. One obvious evaluation technique would be to see if people spend more time reading news with URN or if they read more articles with URN. Actually these evaluations would not be very useful because they do not accurately test what URN is designed to do. It may be that a user of URN ends up reading more articles than they would otherwise because URN only shows them articles they are interested in. With these ideas in mind, we propose two ways to evaluate the URN project.

We have claimed that URN will reduce the amount of noise that users will read and increase the number of useful articles they will read. To test this hypothesis we need a way to determine if people like what they are reading; if they like most of what they are reading then URN is succeeding. This kind of relevance feedback is crucial to the evaluation of the system, but it is important that the evaluation be as painless as possible for the user. If the user is forced to go out of her or his way to provide feedback, then she or he is less likely to do so. We need to give the users an incentive to provide this relevance feedback. The device we will use is the user's choice of which articles to archive. If users are reading mostly noise, then they will obviously not archive very much information, but if they are archiving a large percentage of what he or she sees, then the system is working as planned. The easiest way to evaluate this archival percentage will be to see if it increases over time spent using the URN system. If the system is generating appropriate weighting functions then the archival percentage should increase as the user spends more time using the system. Another possibility is to compare the percentage archived during the use of a conventional newsreader to the percentage archived while using URN, but this would be more difficult as archival procedures are not standardized in conventional newsreaders.

We can also evaluate URN through analysis of the super archive discussed previously. As we said before, the super archive is a conglomeration of many users' personal archives. We have hypothesized that this super archive represents a condensed form of Usenet that removes much, if not all, of the noise. Stated more concretely, we claim that this super archive (given a large enough sample of users) is equivalent to the entire

signal of Usenet. We can test its effectiveness by maintaining two archives of (a subset of) Usenet: one unabridged, and one condensed (obtained from the super archive). We will then allow users to make queries to both the unabridged and the condensed archives. If our claim is correct, then the condensed archive should generate all the same results, but without all the 'false positives' generated by the unabridged version. We can compare the number of 'true positive' results from both archives and thereby evaluate the hypothesis.

5 Current Status

5.1 Research Methodology

At the time of this writing, the research is in its initial phase. The actual system is in the initial stages of implementation (see Section 5.2). Since the system is not yet functional, there are no users and therefore no data has been taken and no evaluation has been performed.

5.2 Research Tool

The URN system is in what we call a 'proto-prototype' stage. The author is still learning about and adjusting to the Egret/Emacs toolset, so the initial release will have almost none of the requirements implemented. This initial release only performs the very basic requirements for needed by any application claiming to be a newsreader: inputting of articles into the Egret HyperBase, reading of articles in the database, following links between articles. A design level description of the current implementation, automatically generated by the Designbase system [Kavoori, 1993], is available as a separate document.

6 Future Directions

The URN system has been designed to enhance the collaborative utility of Usenet, but once it has been shown effective in the realm of Usenet there is no reason why it could not be applied to other areas of electronic information. For example electronic mail could be incorporated into the system quite easily, especially because the format of Usenet articles [RFC-1036] is a superset of the format of Internet electronic mail [RFC-822]. To keep email from getting lost in Usenet, a weighting function could be

created that gives mail a high weight, causing it to rise above most Usenet articles, yet still allow an email message to be prioritized among other email messages. Other information servers such as the WAIS project[5] use weighting functions (though user generated) to determine which information entities to display, so integration with WAIS could be possible.

For the most part this project has concentrated on reverse-engineering techniques to impose more structure on Usenet. We focused on this because the number of Usenet readers and posters is very large making any attempt at imposing structure at the source of postings difficult. The Usenet system is changing and standards groups such as the Internet Engineering Task Force come up with new standards for the format of Usenet. If the ideas in the URN project are shown to be useful, it might be possible to impose more structure at the source by incorporating ideas from URN into the next generation of Usenet posting software.

References

[Johnson, 1991] Philip Johnson. The EGRET project: Exploring open, evolutionary, and emergent collaborative systems. In *Proceedings of the 1991 ECSCW Developer's Workshop*, September 1991. Also published as Technical Report CSDL-91-03, University of Hawaii Department of Information and Computer Sciences.

[Johnson, 1992] Philip Johnson. Supporting exploratory CSCW with the EGRET framework. In *Proceedings of the 1992 Conference on Computer Supported Cooperative Work*, November 1992. Also published as Technical Report CSDL-92-01, University of Hawaii Department of Information and Computer Sciences.

[Kavoori, 1993] Kiran Kavoori. Designbase project status. Technical Report CSDL-TR-93-05, University of Hawaii, Department of Information and Computer Sciences, May 1993.

[5] Information available via ftp at `quake.think.com:/pub/wais/doc`