The following paper was originally published in the
Proceedings of the USENIX 1997 Annual Technical Conference
Anaheim, California, January 6-10, 1997

Experience with GroupLens: Making Usenet Useful Again

Bradley N. Miller, John T. Riedl, and Joseph A. Konstan
Department of Computer Science
University of Minnesota

Experiences with GroupLens: Making Usenet Useful Again*

Bradley N. Miller
John T. Riedl
Joseph A. Konstan

Department of Computer Science
University of Minnesota
email: {bmiller,riedl,konstan}@cs.umn.edu

Abstract

Collaborative filtering attempts to alleviate information overload by offering recommendations on whether information is valuable based on the opinions of those who have already evaluated it. Usenet news is an information source whose value is being severely diminished by the volume of low-quality and uninteresting information posted in its newsgroups. The GroupLens system applies collaborative filtering to Usenet news to demonstrate how we can restore the value of Usenet news by sharing our judgements of articles, with our identities protected by pseudonyms.

This paper extends the original GroupLens work by reporting on a significantly enhanced system and the results of a seven week trial with 250 users and over 20,000 news articles. GroupLens has an open and flexible architecture that allows easy integration of new newsreader clients and ratings bureaus. We show ratings and prediction profiles for three newsgroups, and assess the accuracy of the predictions.

1 The Problem with Usenet Today

1.1 Problem Statement

The information super-highway promises to deliver more information more rapidly than was ever before possible. However, many of us are already overwhelmed with the amount of information we must process each day. The problem of information overload leaves us unable to keep up with the information we need. To long-time readers of Usenet news this problem is especially evident and has caused many users to abandon Usenet altogether. How did we get into this predicament?

The Internet was born in the 1970's by a group of like-minded scientists who used the net primarily to serve the interests of research and academia. This community of on-line pioneers thrived for about 20 years until the rush to the net began in the early 1990s. With this rush came millions of new users with interests that went way beyond the research questions that tied the early community together. Not only did the new users increase the volume of information on the net, they fundamentally changed the culture. What once felt like a small community now feels like a loud impersonal city.

The current estimate of Usenet volume is 21 million users posting 130,000 articles per day. This is up from an estimate of 10,000 articles per day in January 1994. The growth of the Web is even more phenomenal with current estimates that the the size is doubling every 4 months, in terms of both traffic and the number of sites.

The GroupLens project seeks to alleviate the problem of information overload by applying collaborative filtering techniques to Usenet news and other Internet resources. In so doing we hope to help restore order to Usenet,

*Thanks to AT&T Research for their generous support.

and build a renewed sense of community. In an earlier paper [10] we reported on the initial GroupLens architecture and a small scale pilot test with approximately 12 participants at our local universities. In this paper we report on the new GroupLens architecture, including the newly published open protocols, and a larger scale Internet wide user test involving more than 250 participants and tens of thousands of ratings. The next sections describe some of the past and current strategies for fixing Usenet. Section 2 presents the GroupLens server architecture. Section 3 discusses the client library and the adaptation of newsreaders to support GroupLens. Section 4 presents data gathered and lessons learned from the seven week user trial, and section 5 discusses future work and presents some conclusions.

1.2 Non-collaborative Solutions

Since the early days of Usenet people have tried to find ways to reduce the number of messages they must process each day. In this section we consider *non-collaborative solutions*, which are solutions that use only information from a single user. Some of the earliest non-collaborative techniques relied on matching keywords in the header fields of the news articles. Later, more sophisticated keyword matching techniques emerged that applied techniques from information retrieval.

Kill files / Score files One of the first methods introduced to the Usenet community to reduce the noise level was the killfile. Recently scorefiles have been introduced as a more general mechanism. A killfile allows a user to specify certain subjects or authors that he never wants to see, while a scorefile allows the user to give interesting subjects and authors high scores and uninteresting subjects and authors low scores [5]. The problem with these techniques is their coarseness. Not all articles containing a desired keyword are interesting, and even generally poor writers occasionally produce an article worth reading. Additionally, keywords are difficult to identify in the presence of aliases, synonyms and misspelled words.

Moderated Newsgroups Another approach to reducing the noise level on Usenet is the creation of moderated newsgroups. In a moderated newsgroup one person, the moderator, must approve each article before it is distributed throughout Usenet. The moderator is responsible for rejecting articles that are off-topic, inflammatory, or generally of poor quality. The problem with moderated groups is that they require a large time commitment from the moderator, and the quality judgment is left up to a single person.

Programmable Agents Programmable agents are simple programs that perform actions on behalf of users. Programmable agents have been used in information filtering to prioritize messages, gather messages into folders by keyword, or even reply to messages. For instance, the Information Lens system enables even unsophisticated users to automatically perform actions in response to messages [6]. Object Lens extends the Information Lens to other domains, including databases and hypertext [3].

Intelligent Agents The July 1994 issue of *Communications of the ACM* is devoted to the state of the art in intelligent agent research. This issue includes discussions of several agents designed to reduce information overload. [4]. The agents address meeting scheduling, email handling, and netnews filtering. The netnews agent is known as NewT [14]. A user trains NewT by showing it examples of articles that should and should not be selected. The agent performs a full text analysis of the article using the vector-space model [11]. Once the agent has gone through initial training it starts making recommendations to, and accepting feedback from the user. Based on user feedback NewT is able to make weighted judgments about news articles containing keywords. Intelligent agents for information filtering suffer from the same drawbacks as keyword based techniques. An additional problem is that agents must be trained. Norman points out in [9] that interaction with and instruction of agents is a difficult problem that has not been solved satisfactorily.

1.3 Collaborative Solutions

Collaborative filtering systems make use of the reactions and opinions of people that have already seen a piece of information to make predictions about the value of that piece of information for people who have not yet seen it. Collaborative filtering is already used heavily in informal ways. Users regularly forward articles, or references to articles, to their friends and colleagues with the explicit or implicit message: "You will like this." Collaborative filtering systems atttempt to formalize this process to more effectively incorporate a larger set of users and data. The utility of collaborative filtering extends beyond the domain of Usenet news into the realm of movies, videos [2], and audio CDs [13].

Usenet represents a uniquely challenging problem for a collaborative filtering system because of the sheer volume of information items for which ratings must be collected and predictions calculated. The 130,000 new messages produced on Usenet each day dwarfs the number of new CDs and movies produced in an entire year.

Tapestry The Tapestry system [1] is an early collaborative filtering system designed to help small groups of people work together to solve the information overload problem. Tapestry makes sophisticated use of subjective evaluations. It allows filtering of all incoming information streams, including email and Usenet news. Many people can post evaluations and users can choose which evaluators to pay attention to. The evaluations can contain text, not just a numeric rating or boolean accept/reject. In the Tapestry system users can combine keyword criteria, along with subjective criteria to form requests. An example request might be "Give me all the articles containing the word *collaborative* that Pat has evaluated and where the evaluation contains the word *excellent*." Tapestry works well in a close-knit community with common interests. GroupLens extends the concepts in Tapestry system in two ways. First, GroupLens provides predictions based on the aggregation of ratings entered by other users. Second, GroupLens does not require the user to know whose evaluations to use in advance.

NoCeM NoCeM (no see 'em) is a system that makes it possible for anyone to attempt to cancel an article that is widely cross-posted or seen as a blatantly commercial posting. Under the NoCeM model, any person on the net who sees something they think shouldn't have been posted can issue a NoCeM notice. However, just as with any other type of Usenet message, the weight the notice carries will be no greater than the poster's net.reputation. If people agree with the issuer's criteria and also feel that this person is a good judge of that standard then they will accept his/her notices. When a NoCeM notice is accepted by a user it will typically mark the message as read in the user's newsrc file. NoCeM notices could instead be used to remove the message from the local spool, thus keeping all users on the local system from seeing the article.

1.4 The GroupLens Approach

GroupLens is a collaborative filtering system for Usenet news. The aim of GroupLens is to help people work together to find articles they will like in the huge stream of available Usenet articles. In effect, GroupLens automatically selects for you a group of people to act as your personal moderators for a given newsgroup. These moderators are selected by finding people with whom you have had substantial agreement on past articles. Users can ensure their privacy by entering ratings under pseudonyms without reducing the effectiveness of the predictions.

Usenet news readers can take advantage of

GroupLens by reading news with a GroupLens-aware news client. We provide several clients and make a library available for newsreader authors who want to make their newsreaders GroupLens-aware. An example client is shown in figure 1. The newsreader connects to the user's local NNTP server to retrieve Usenet news articles, and it also connects with the GroupLens server to share filtering information. Whenever the user fetches articles from a newsgroup, the news reader sends a message to the GroupLens server requesting *predictions* of how the user will value each article. Those predictions are displayed alongside the article titles as a bar. When reading news articles, the user may enter ratings of the actual value of each article. Those ratings are sent back to the GroupLens server to serve as input for other users' predictions and to update the correlations between this user and other users. The more one uses the system, the more data there is upon which to base predictions.

We believe that GroupLens provides the best opportunity for managing the overwhelming amount of data in Usenet news. In addition to the general benefits of collaborative filtering over non-collaborative solutions, GroupLens has a scalable open architecture that can support a large number of users and data elements. The GroupLens architecture also can support a variety of algorithms for collaborative filtering, allowing system designers to trade off between efficiency of calculation, storage requirements, and the degree of personalization of predictions.

2 The GroupLens Architecture

2.1 Overview

At the heart of GroupLens lies the GroupLens Ratings Bureau (GLRB) which functions as a request broker for the distributed collaborative filtering engine. To implement the request broker we have adopted a "process pool" model that allows the GLRB to identify incoming requests and hand them off to the appropriate background daemon. To keep the number of network connections low we have adopted a virtual session model for each client connection. Once a client as logged in the client is given a session token that is valid until the client logs out or becomes inactive.

In figure 2 we see two newsreading clients in different stages of communication with GroupLens. The `tin` client is in the process of establishing a connection with the GLRB; its request has not yet been determined. On the other hand the GLRB has already assigned the `xrn` client to the appropriate prediction daemon.

The filtering engine contains four primary modules. A prediction module, a ratings module, a correlation module, and a data management module. The GroupLens architecture is open, and the modules communicate with each other through a well defined protocol. This allows any of the modules to be replaced by a functionally equivalent module so long as the new one conforms to the protocol.

The primary goal of the collaborative filtering engine is to provide clients with accurate predictions quickly. Predictions are calculated by one of the daemons in the prediction daemon pool. To calculate a prediction for an article the prediction daemon requires two inputs: a measure of similarity between pairs of users, and ratings for the article in question. The way in which these two inputs are combined for each prediction is described in [13, 10, 7]. Our performance goal for the the prediction algorithm is to be able to calculate and deliver 100 predictions in less than two seconds. In practice we are able to deliver 100 predictions in 4.2 seconds on a Sparc 5 workstation with 32Mb memory, running Solaris 2.4.

Pairwise similarity between users is determined by the correlation program. Similarity between two users is determined by how they have rated articles in the past. Because user's correlations change relatively slowly, and because newsreading tends to be a daily activity, the correlation program is run once a day.

Ratings for Usenet articles are received in

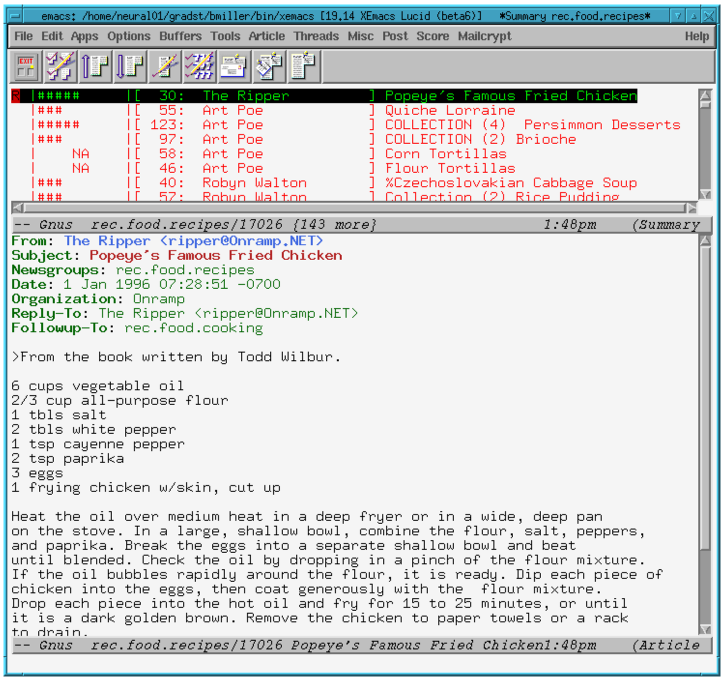

Figure 1: The GNUS newsreader with GroupLens predictions

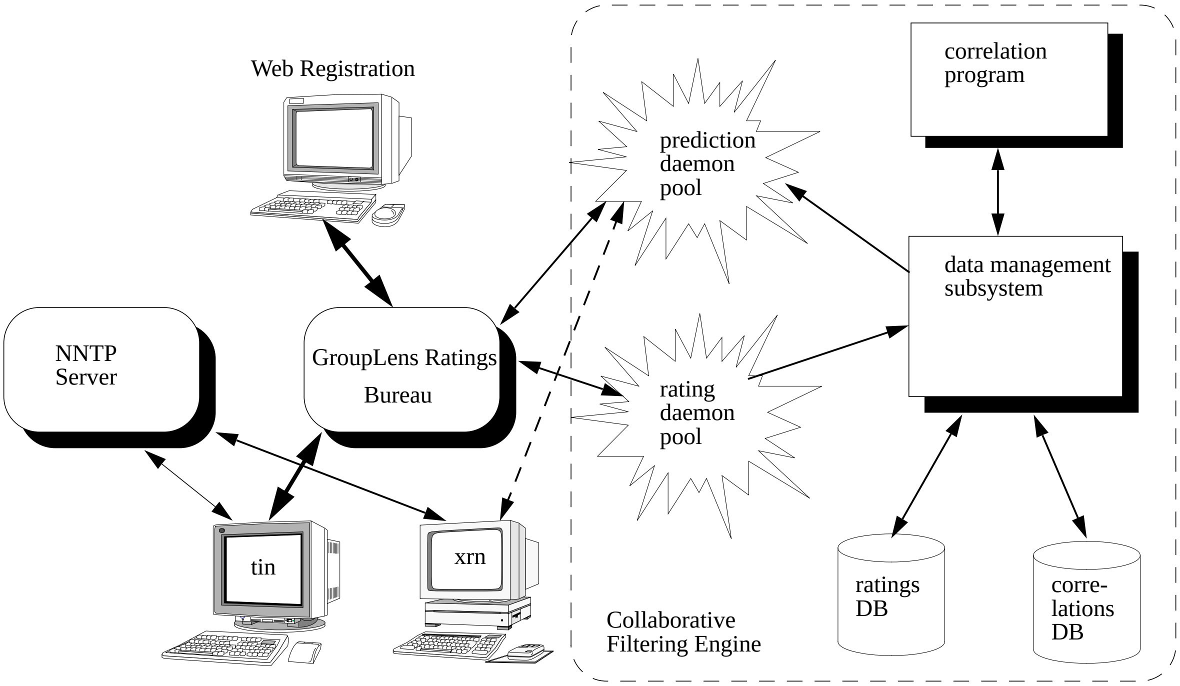

Figure 2: GroupLens Architecture Overview

batches from the newsreader client as defined by the GroupLens protocol. It is the responsibility of the ratings daemon to receive ratings from the clients as fast as possible and to ensure that the ratings are eventually stored in the ratings database. Our performance goal for ratings is to be able to accept 100 ratings in less than 1 second. In practice we are able to accept 100 ratings in less than 0.5 seconds.

The data management subsystem is responsible for maintaining both the ratings and correlations databases. Logically you can think of the ratings database as a large matrix organized with message-ids indexing the columns, and pseudonyms indexing the rows. The current database for the rec.humor newsgroup has 20,837 columns and 74 rows for a total of 1,541,938 cells. However, only 30,537 or 0.01% of these cells are occupied. We have devised a storage mechanism that minimizes access time for an individual cell, and minimizes the space required to store the matrix.

We have designed our data management interface so that we can plug in one of several DBMSs on the back end, while maintaining a consistent interface on the front. Currently we support three back ends: gdbm [8], Illustra, and OBST.

2.2 Protocol

The glue that ties all of the modules together, and allows the newsreading clients to talk to the GLRB and other modules, is the GroupLens protocol. The protocol consists of five major commands. We'll give a brief overview of the major commands here. The details of the protocol are available on-line [15].

Three key concepts in the protocol are *pseudonyms*, *tokens*, and *message-ids*. Pseudonyms are the secret identifiers selected by users to identify themselves to the GroupLens system while maintaining their privacy. Tokens are integers returned from the server to represent the state of a logged in user. The server maintains just enough state for each token so the user does not have to authenticate herself each time she requests predictions or submits ratings. The server discards tokens after a timeout period. Message-ids are used by the clients to identify items they wish to rate or get predictions for. Message-ids in the Usenet trial are the standard Usenet message identifiers used by news clients to identify messages to the news server.

Register In figure 2 we see a World Wide Web client talking to the GLRB from the GroupLens Registration page. The format of a register command is `register pseudonym`. When the GLRB receives a register command, it checks the user database to make sure that the given pseudonym does not already exist.

Login The format of the login command is `login pseudonym`. When a login request is received the GLRB checks to see if the pseudonym is valid. If the pseudonym is valid the client is given a session token. To increase security a password may be optionally supplied as part of the login command.

Logout The format of the logout command is `logout token`. When a logout command is received the token is removed from the list of active tokens, and the token number is invalidated. Any future requests using this token number will be refused.

GetPredictions The format of the getpredictions command is `getpredictions token newsgroup`, followed by a list of Usenet message-ids. When the GLRB sees that the request is `getpredictions` it validates the token and newsgroup name, and then passes the request to a free prediction daemon. The prediction daemon reads the list of message-ids and returns either a prediction or a keyword indicating no prediction for each message-id.

PutRatings The format of the putratings command is `putratings token newsgroup`, followed by a list of tuples that contain the message-id and rating. The ratings daemon simply reads the list of tuples and informs the client that it has received them. When it has time, the ratings daemon writes the list to the database.

2.3 Privacy

Privacy is an important issue in a large scale collaborative filtering system. There are three ways to handle user privacy issues in a collaborative filtering system. First, users may be anonymous so that ratings are submitted without any user identification. When ratings are submitted anonymously, the only operations the system can perform are aggregate operations such as the average rating [7]. Second, users may be known to all other users. In this case the ratings are closely associated with the reputation of the rater, and users seeking recommendations or predictions may specify which other users to use in generating predictions [1]. This option requires users to give up the privacy of their ratings. The third option, employed by GroupLens, uses *pseudonyms* to uniquely identify every user. Using pseudonyms allows ratings to be associated with a user, and allows predictions to be customized for users based on their correlation with other pseudonyms. In GroupLens, pseudonyms and their associated ratings are publicly available. However, we do not associate these pseudonyms with a user's real identity and we use an authentication protocol [12] that prevents a user from using another's pseudonym.

3 Filtering Clients

3.1 News Readers

The primary user interface for the GroupLens system is a set of newsreaders that are adapted to use the GroupLens server as well as the local NNTP server. We currently support three Unix-based newsreaders: `xrn`, `tin`, and gnus. We are in the process of adapting newsreaders for the PC and Macintosh platforms.

To simplify the process of adapting newsreaders, we have implemented and freely provide a GroupLens client library. This library handles all GroupLens server communication, manages local configuration files, and provides data structures to simplify the integration of ratings and predictions into an existing news reader. When using the library, the newsreader author is freed from the details of logging into the GroupLens server and maintaining a token, and from the details of the GroupLens protocol.

Adapting a newsreader to use GroupLens involves three steps:

1. Passing the set of article IDs to the client library (to retrieve predictions) when retrieving article headers for a newsgroup.

2. Recording article information, including user ratings, in the GroupLens article table, and calling the library routine to submit these ratings after finishing each newsgroup.

3. Defining a user interface for displaying predictions (some support is in the library) and for receiving ratings from users.

Our experience modifying newsreaders has shown that the interface changes are the hardest part of the process. Many newsreaders use nearly every key on the keyboard, and consequently require creative interface design to maintain a consistent interface. While displaying predictions was somewhat simpler, we have found that some news reading models are not as amenable to selection by title and prediction, and accordingly plan to investigate methods for providing summary predictions for threads (perhaps at the client interface).

We were able to add GroupLens support to `xrn` with less than 1000 additional lines of code. These lines represent less than 3% of the total `xrn` source code.

3.2 Filter Bots

We use the name *filter-bot* to refer to simple filter programs that algorithmically ("robotically") supply useful information to a filtering system. In GroupLens, a filter-bot is a program that assigns a rating to a Usenet article based on some simple computable criteria. Filter-bots are implemented using the client library and enter ratings under their own unique pseudonyms. This allows real users to weigh the ratings of the filter-bots along with ratings from other users. Examples of filter-bots we have implemented include an article length filter-bot and an excessive quoted text filter-bot. We intend to implement several more including a reading level filter-bot[1], a prolific author filter-bot, and an excessive cross-posting filter-bot.

Filter-bots measure syntactic features of articles, providing additional ratings with which users can correlate, potentially improving predictions. Filter-bots facilitate incorporating new information filtering algorithms into the GroupLens architecture. Filter-bots also mitigate the *first rater problem*, which stems from the fact that in order to compute a prediction for an article at least one previous rating must be available. Filter-bot ratings for an article can be computed immediately, so they are always available for users.

4 Experiences

We now turn to the results of a user trial we conducted to test the GroupLens architecture. The user trial began February 8, 1996 when we posted an announcement to the comp.os.linux.announce newsgroup. In order to participate in the trial, users had to be willing to use one of the newsreaders that had been enhanced with GroupLens support. The newsreaders available were gnus-5.1 (for emacs), `tin`, and `xrn`. Participants also had to be willing to read and rate articles in one of our supported newsgroups. The groups supported for the trial included the entire comp.os.linux hierarchy, rec.humor, rec.food.recipes, comp.lang.c++, comp.lang.java, and rec.arts.movies.current-films. Participants were told to rate articles according to the following definitions:

1. This article is really bad! a waste of net.bandwidth.
2. This article is bad.
3. This article is neither good nor bad.
4. This article is good.
5. This article is great, I would like to see more like it.

Through the first seven weeks of the trial, we had 250 users register to use GroupLens. The total number of ratings received during the seven week period was 47,569. These ratings are spread over a total of 22,862 distinct messages. Over the same seven week period GroupLens provided over 600,000 predictions to users. This ratio of ratings to predictions is appropriate for a noisy domain like Usenet news, since it suggests that ratings help users choose which items to review.

In the next two sections we take a look at the general question, "does it work?" We'll look at this question from two perspectives: First, do the predictions accurately reflect what the user rated the article? Second, do users find the predictions useful, and do they believe them? We used data collected from the earlier GroupLens trial [10] to develop prediction algorithms, and evaluated the performance of the prediction algorithms based on the data from the present trial. The algorithms were not changed during this trial.

4.1 Accuracy of Predictions

Our experience has shown that the prediction program behaves differently for different newsgroups. To study this point, we

[1]A reading level filter-bot rates articles according to the minimum grade level for which the vocabulary and sentence structure would be appropriate.

will examine the accuracy of the prediction program for three representative newsgroups. `rec.humor`, `rec.food.recipes`, and `comp.os.linux.development.apps`

For each of the newsgroups we will compare the accuracy of predictions for two different ways of calculating the predictions. We will calculate a personalized prediction for each user for each article using the Pearson coefficient as a similarity measure between users, as described in [10]. For comparison we will calculate the average rating entered for each article. We compare each prediction against the actual ratings entered by the users. It is useful to look at the average because it is fast to calculate and requires very little storage. On the other hand, the average does not allow for any personalization of the ratings.

The metrics we will use to measure the accuracy of the algorithms include the *mean squared error* $\overline{E^2}$; The *mean absolute error* $|\overline{E}|$; the *standard deviation*, σ, of $|\overline{E}|$; and the Pearson correlation coefficient r between ratings and predictions.

To measure the mean absolute error we take the absolute value of the difference between the actual rating entered by the user and the prediction computed by the algorithm for each rating/prediction pair, and compute the mean of all of the differences. The lower the mean absolute error, the better the algorithm. The standard deviation of the error is a measure of how consistently accurate the algorithm is. One problem with using the mean absolute error is that it does not sufficiently penalize algorithms that make large errors. The mean squared error, like least squares regression, disproportionately penalizes algorithms that make large errors more than small. We want to penalize large errors because users probably don't distinguish between a prediction of 1.5 and 2.0. On the other hand, users will notice if an algorithm predicts something to be a 4.0 that should really be 2.0.

As mentioned in section 2, the prediction algorithm requires a measure of similarity between pairs of users (correlation), and ratings. The nature of the newsgroup appears to have an effect both of these factors. In figure 3 we show the rating profiles for all groups combined, and for the three newsgroups.

In `rec.humor`, 83% of the ratings are 1 or 2. This reflects the paucity of funny articles and the overabundance of name-calling, flaming, and completely silly discussions of World War II. `Rec.humor` is a good example of a newsgroup where there is a clear metric for determining a rating: "Is it funny?" The fact that there is a clear metric for judging each article, and the fact that there is so much noise leads to a high level of correlation between pairs of users. This is illustrated in figure 4 where we can see that most pairs of users have a high positive correlation.

method	$\overline{E^2}$	$\lvert\overline{E}\rvert$	σ	r
average	1.1	0.63	0.88	0.49
personalized	0.94	0.67	0.68	0.62
all-ones	2.01	0.78	1.18	NA

Table 1: Summary of Results in Rec.humor

In table 1 we see the comparison between average and personalized predictions for `rec.humor`. Because there is such a high degree of correlation between users, we see that the average is slightly better than the personalized algorithm in terms of the mean absolute error. One might think that given the ratings profile for rec.humor the best strategy to minimize error would be to simply predict 1 for every article. The row called "all-ones" in table 1 shows that this is not a good strategy after all.

In `comp.os.linux.development.system`, and `rec.food.recipes` we see that the ratings are more evenly distributed (see figure 3). `Rec.food.recipes` is a moderated newsgroup, so all of the posts are on topic, and there is no name calling or spamming. In addition users once again have a clear metric for rating an article: "Would I like to cook this?" However, as figure 4 shows, users in `rec.food.recipes` have a lower correlation. The reason for this is that ratings are based literally on taste. For

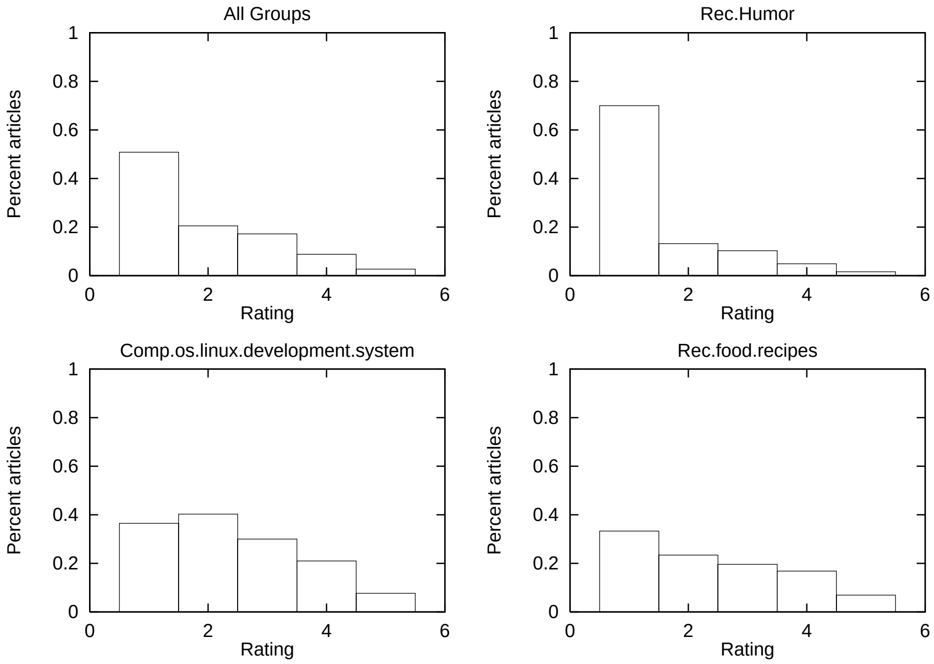

Figure 3: Summary of ratings in selected newsgroups

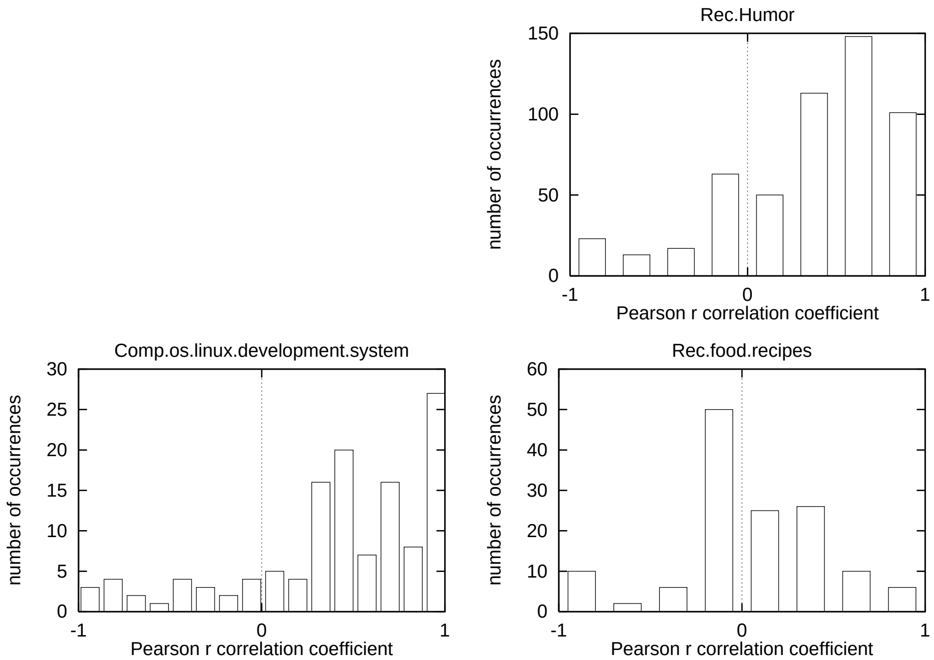

Figure 4: Summary of correlation between users

example two users may agree all the time on desert recipes, but one may be a vegetarian who rates all recipes with meat low, and the other may be a carnivore who rates recipes with meat high.

In table 2 we summarize the results for `rec.food.recipes`. We see that the errors in this group are uniformly higher than for `rec.humor`. This can be attributed to the fact that the correlation between users is low for this newsgroup. However, we do see that the personalized predictions are better than average predictions for all of our error metrics.

method	$\overline{E^2}$	$\lvert\overline{E}\rvert$	σ	r
average	2.65	1.29	0.97	0.05
personalized	1.94	1.09	0.86	0.33

Table 2: Summary of Results in Rec.food.recipes

Determining what to rate an article in `comp.os.linux.development.system` is more difficult than either of the two previous newsgroups. When rating an article in this group a user must weigh several factors:

- Is the article appropriate for this newsgroup?
- Is the topic of the article interesting to me?
- Is the article well written?
- Is the article factually correct?

Despite all of these factors the readers of `comp.os.linux.development.system` have a high degree of correlation. This may be because the early adopters of the GroupLens system are all likely to be fairly sophisticated linux users. In table 3 we see that the personalized predictions are again more accurate than the average.

4.2 Effect of Predictions on Users

We'll now look at what effect, if any, the predictions have on a user's likelihood to read and rate a message. One result is clear: users are more likely to rate messages if they see that they are also getting predictions. Further, our analysis of the rating patterns of users shows that users are almost twice as likely to rate an article for which they see a prediction greater than three than they are for an article with a prediction of three or less.

method	$\overline{E^2}$	$\lvert\overline{E}\rvert$	σ	r
average	1.28	0.78	0.82	0.41
personalized	0.91	0.71	0.64	0.55

Table 3: Summary of Results in Comp.os.linux.development.system

4.3 Bootstrapping

One rather important lesson that we have learned over the course of this project is the difficulty of bootstrapping collaborative filtering in Usenet newsgroups. While we expected some inertia among users, and in particular recognized the difficulty in asking users to upgrade or change newsreaders, we were surprised by some of the social difficulties involved in bringing collaborative filtering to Usenet news.

Collaborative filtering must have users to be useful. In fact we believe that a collaborative filtering system has built in incentives that encourage more people to participate. To ensure value for trial users, we made an effort to add GroupLens support for newsgroups slowly, and only after ensuring that we had at least one or two active reader/raters who would keep the group going. We would then post a message to the newsgroup itself, inviting others to use GroupLens and pointing them to our web page for registration and software. It was here that we ran into a very tricky bootstrapping problem.

Discussing GroupLens was off-topic for almost every newsgroup we encountered. Accordingly, our messages were often ignored, and there was no follow-up discussion within the group. In retrospect, this is not surprising. A posting about a better way to read news is not funny (and therefore does not belong in `rec.humor`), it is not about linux software and development (and therefore does not belong in `comp.os.linux.*`), and similarly is out of place in the very newsgroups where we expected it to be useful. Few Usenet news readers choose to read newsgroups *about* reading news.

We did get a user base, albeit more slowly than we would have liked. We have since recognized more effective ways of bootstrapping: working more closely with newsreader maintainers to become part of the standard distribution, providing limited service for a wider range of newsgroups to create more general interest, and direct promotion through demonstrations and other publicity.

5 Conclusions and Future Work

The GroupLens trial has demonstrated the efficacy of collaborative filtering for Usenet news. We have learned about the challenges of this vast domain. Each newsgroup brings forward new characteristics that affect the accuracy of our predictions. The sheer volume of Usenet news has forced us to have an efficient implementation. The numbers are in, and GroupLens provides value to participants. Anecdotal evidence supports this conclusion as we hear from users who long-ago abandoned the `rec.humor` newsgroup returning to it with GroupLens guiding them to a handful of funny articles in just a few minutes each day. Still, our work is far from finished.

There are many areas of future work to reduce the possible costs to users of using collaborative filtering. These costs come in three forms: (i) time spent entering a rating; (ii) performance costs incurred by the GroupLens software; (iii) the time wasted in reading articles that are predicted to be better than they really are.

One way of reducing the time spent entering a rating is to rely on implicit measures of interest, such as how long you spend reading an article, or whether you print or file the article after reading it. Collaborative filtering studies are needed to compare the costs and benefits of explicit ratings with implicit ratings.

Performance costs for the current GroupLens system are low, but these costs increase with the number of users. One way to keep performance costs low is to develop distributed GLRBs that perform correlation and prediction independently on subsets of the user population. Scaling GroupLens to the Internet will require distribution, to keep communication and computation times.

Time wasted reading articles with high predictions that are actually uninteresting is difficult to control. In practice, predictions become increasingly accurate as more readers rate the the article. One way to make it easier for users to use collaborative filtering to select quality articles would be to develop prediction algorithms that include a confidence measure for the accuracy of the prediction. Confidence measures would help users make the tradeoff between opportunity cost and expected value.

GroupLens is an open architecture with freely distributable protocols. Anyone who wants to participate can write a news client to connect with our GLRB, or a new GLRB that offers improved service to our news clients. An open architecture encourages interaction and innovation by the community. For instance, one GroupLens participant has already written an proxy GLRB that downloads his ratings to Poland overnight so he gets better interactive performance!

The client library further encourages participation, by simplifying the task of integrating GroupLens with new news clients. For most news clients, only 2-3% of the code must be modified to support GroupLens. Recently, several of the maintainers of popular news clients have announced support for GroupLens. We look forward to working with the community to add GroupLens support to additional newsreaders. We also encourage the development of new filter-bots that use the client library to communicate computed ratings to GroupLens.

Usenet is on the one hand a rich and valuable information resource, and on the other hand a quagmire of the useless and the tasteless. GroupLens lets us team up to drain the quagmire and separate the valuable from the useless. If we work together we can each peruse a fraction of the articles submitted each day, in exchange for having the interesting articles pointed out to us. More participants means more ratings available to GroupLens, which means even better predictions. The GroupLens experience is on-going at `http://www.cs.umn.edu/Research/GroupLens`. Join us in making Usenet useful again!

6 Acknowledgements

Many people have participated in making GroupLens a success. Paul Resnick deserves special recognition for co-founding the project with John Riedl. The HackWeek members – the authors, Jon Herlocker, Dave Maltz, and Paul Resnick – designed and implemented the new architecture. Danny Iacovou, Mitesh Sushak, and Pete Bergstrom worked on early versions of the system.

References

[1] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[2] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI)*, pages 194–201. ACM, 1995.

[3] Kum-Yew Lai and Thomas W. Malone. Object lens: A "spreadsheet" for cooperative work. In *Proceedings of International Conference on Computer Supported Cooperative Work*, Portland, OR, September 1988. ACM Press, New York, NY.

[4] P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40, July 1994.

[5] Lars Magne-Ingebritson. *Gnus 5.0 Reference Manual.* Available at: http://www.miranova.com/gnus-man/gnus.html.

[6] T.W. Malone, K.R. Grant, F.A. Turbak, S.A. Brobst, and M.D. Cohen. Intelligent information-sharing systems. *Communications of the ACM*, 30(5):390–402, May 1987.

[7] D.A. Maltz. Distributing information for collaborative filtering on Usenet net news. Master's thesis, MIT Department of EECS, May 1994.

[8] Philip A. Nelson. *gdbm–1.7 Reference Manual.* Available from: ftp://prep.ai.mit.edu/pub/gnu/gdbm-1.7.3.tar.gz.

[9] D.A. Norman. How might people interact with agents. *Communications of the ACM*, 37(7):68–71, July 1994.

[10] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of Computer Supported Cooperative Work Conference.* ACM SIG Computer Supported Cooperative Work, 1994.

[11] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, 1983.

[12] Bruce Schneier. *Applied Cryptography.* John Wiley & Sons, 1994.

[13] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM Conference on Human Factors in Computing Systems CHI*, pages 210–217, 1995.

[14] B. Sheth and P. Maes. Evolving agents for personalized information filtering. In *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*. IEEE Computer Society Press; Los Alamitos, CA, USA, 1993.

[15] The grouplens home page. http://www.cs.umn.edu/Research/GroupLens.