Research Report

Recommender Systems: Knowledge from Mining User Experiences

Stephen C. Gates, Charu C. Aggarwal
IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598

Paul P. Maglio
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120

Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

Recommender Systems: Knowledge from Mining User Experiences

Stephen C. Gates[1], Charu C. Aggarwal[1], and Paul P. Maglio[2]

[1]IBM T. J. Watson Research Center, Hawthorne, NY 10532 USA

[2]IBM Almaden Research Center, San Jose, CA 95120 USA

Abstract

One largely untapped source of knowledge about large data collections is contained in the cumulative experiences of individuals finding useful information in the collection. Recommender systems attempt to extract such useful information by capturing and mining one or more measures of the usefulness of the data. This paper considers the architecture of recommender systems in general, and offers a specific example called SurfAdvisor to illustrate how some design issues inherent in such systems can be addressed. Specifically, our system recommends useful documents to World Wide Web surfers. It uses a series of asynchronous processes managed by an intelligent proxy to (a) classify each Web page, (b) allow users to vote on the usefulness of the page, and (c) provide four types of recommendation on any of 1167 subject areas. In addition, our system also provides users with title, abstract, voting, and speed information about each link on a Web page, all in real time. In the end, we show that appropriate mining of data by a well-designed recommender system can significantly improve the perceived usefulness of the Web.

1. Introduction

One approach to knowledge discovery is that taken by *recommender systems* [14] or collaborative filtering systems [7], which define *useful* in a social way, mainly by the collective experience of others. For example, recommender systems have been used to suggest (a) movies based upon reviews of movie viewers [20], (b) Web pages based upon citations in Usenet articles [15], and (c) books based upon purchase patterns of book buyers [19]. Most of these systems attempt to tailor recommendations to a given individual by matching user interests with like-minded others [19, 20].

An underlying premise of such systems is that the collective experience of other users is a form of knowledge that is worth mining. In fact, recommender systems are a logical extension of the social networks that people use in obtaining new information. Given a problem outside of their area of expertise, people often turn to those around them to get the information, and if that doesn't work, they ask their friends or colleagues for the names of others who might be experts. One interesting kind of recommender system, in fact, automates the process of discovering this sort of social network on the Web [11].

Automatic recommender systems are particularly useful for problems that are too large to be solved by consulting the people in ones own social network. Asking a friend for a movie recommendation may be better than asking a recommender system; however, if your movie tastes don't coincide with those of your friends or run to obscure movies, then a recommender system might give better recommendations. For many problems, no small set of friends or colleagues can know the answer; for example, to asking "what are the best 10 pages on the Web on each of these 100 topics." This question is unlikely to be answered well by even a very Web-savvy friend.

In this paper, we present SurfAdvisor, a recommender system for Web pages. To provide recommendations, SurfAdvisor relies both on knowledge gathered from users on the utility of sites they visit as well as on automatic analyses of the structure and content of the Web. Using SurfAdvisor as an example, we will argue that recommender systems are perceived as valuable to the extent that they:

(1) Deal with large-scale problems,

(2) Contain recommendations covering a significant fraction of available objects,

(3) Customize the recommendations so that they appeal to all users,

(4) Offer recommendations that are perceived as trustworthy,

(5) Offer timely recommendations, and

(6) Require little effort by the user.

To do this, we first outline a general architecture for recommender systems, discussing the design of each component in general terms to show the variety of ways in which these objectives can be met. We then discuss the details of our SurfAdvisor implementation to illustrate design concepts and tradeoffs. In the process, we also discuss the performance of our system and suggest ways in which it might be improved.

2. Generalized Recommender Systems

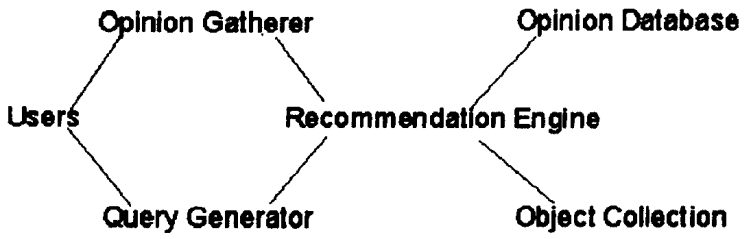

Figure 1.

In its most general form (see Figure 1), a recommender system contains an *opinion gatherer*, which gathers opinions from some source (often but not always system users) and stores them in an *opinion database*. When a user or *query generator* creates a query, one or more objects are found in an *object collection*; the objects are selected based upon both the user's query and the database of user opinions. A *recommendation engine* performs the selection, based upon a set of criteria (e.g., the correlation between the opinions of the user and other users already in the opinion database, versus the items in the object collection). A user might ask, for example, "show me the most useful documents on skiing", and the recommendation engine might select those documents on skiing that previous users had found most useful.

We now discuss each of these components in turn.

2.1 Opinion Gatherer

The key to the recommender system is its *opinion gatherer*, which collects user opinions about the object collection. This is often an asynchronous, background process, although it need not be so. For example, the opinion gatherer might mine a database for references to the object collection (e.g., the system that peruses all Usenet articles for references to Web pages [15]). Or it might gather opinions from the users as they use the system (e.g., systems that collect information about which Web sites a user visits [17], how long or how often a Web site is visited, or how a user feels about a given item [20]). Other alternatives include gathering opinions from one or more select opinion-makers (e.g., the experts hired by Yahoo! to evaluate Web pages [23]) or specialized monitoring systems (e.g., the home television monitoring by television rating companies) or even celebrity ratings. In any event, the main factors in designing an opinion gatherer are social rather than technical, and include the privacy of the user, the ease of collecting the rating, and the rewards for giving the rating.

2.1.2 Opinions Collected from Named Individuals

Because ratings are considered more credible when they are collected from individuals known to the user, or at least from named individuals, most recommender systems utilize recommendations collected from other users. To do this, four major issues must be confronted:

(1) Privacy. Privacy is of significant concern to many individuals when opinions are collected. Opinion gatherers must implement a policy that specifies a) the circumstances under which a person's opinion will be gathered (particularly whether it requires an overt act by the user), b) whether the opinion will be tagged in any way with the user's identity, c) the security of the database in which the opinions are stored, d) how broadly a person's opinion can be disseminated, e) when the opinion should be presented as anonymous or pseudo-anonymous, f) who can access the opinion data base, and g) whether users can contact one another (e.g., in chat groups created based upon affinity groups discovered by the recommender system). Existing recommender systems implement a wide range of privacy policies [18, 19, 21]. One simple

possibility is to maintain anonymous opinions, but the behavior of completely anonymous raters may end up being quite destructive [6].

(2) Ease of rating. One major impediment to the success of any recommender system is the difficulty in getting users to share opinions. Most systems try to make the collecting of opinions as easy as possible. Some systems collect existing public ratings (e.g., the mentions in Usenet postings [15]), or automate the sharing of private ratings (e.g., automatic bookmark sharing within a workgroup [9]). Other systems do this by watching the user and recording how often or for how long they view a given document [17]; this may raise privacy concerns, however.

(3) Rewarding the user. Most recommender systems reward users by giving them information (e.g., recommendations about movies they might like to see [20]). However, in systems in which the user is required to vote or take other overt acts, additional motivation may be required. As Resnick and Varian have pointed out, however, the first recommender system which becomes widely successful may be able to offer much higher quality recommendations than systems without a large opinion database [14], so the issue of rewards may become more important in the future.

2.1.3 Tradeoffs in Building Opinion Gatherers

Ultimately, social factors must be translated into a specific software design. The key factors are the size of the database and the relationship among the users. For a small object collection` (e.g., a few thousand movie titles), it is possible to have one or a few persons examine and rate the entire collection` , particularly if the collection is relatively static. Similarly, if the users are all part of a single workgroup, it may be possible to have one person rate each object and attach the rater's name to the rating. For large object collections (e.g., the Web) and large numbers of users, however, the design decisions are more difficult. In these cases, large numbers of raters may be required to rate a significant fraction of the objects. Hence, the chances of the raters being known to the users decrease, and the credibility of the ratings may be more open to question.

A second set of tradeoffs is a bit more subtle: how many different kinds of opinion gatherers are needed? Current systems typically gather opinions from only one class of source: users, experts, or a database of previously gathered opinions. However, future systems will probably gather from a wide range of sources to give the best coverage.

A third, and potentially more difficult tradeoff involves privacy. Most current systems offer recommendations for free. However, some systems trade that free cost for decreased privacy by sharing user profiles with merchants. This improves the ability of the merchant to target information or services to individual customers. However, particularly in Europe, where information sharing is a substantial legal concern, these schemes must be very well thought-out.

2.2 Query Generator

The *query generator* creates requests for objects from the recommendation engine. For instance, in many existing systems, the user is given a series of sample objects to rate, and the query generator effectively asks the recommendation engine to provide objects based upon the user's set of ratings. If the user has not yet provided a sufficient number of ratings, the recommendation engine simply retrieves another sample object. Another type of query generator is a standard keyword-based query, such as those as used by Web search engines.

A query generator might be triggered by direct input from the user, either via natural language or via voice recognition, and might be triggered only when the user asked for something using words like "best" or "most useful." Or a query generator might trigger on specific events, such as a specific time or date or browser event or device activation (e.g., turning on your television might trigger it to query a recommendation engine for a set of movies or TV programs to suggest).

2.3 Recommendation Engine

The *recommendation engine* responds to a query by delivering a document to the user. In the simplest case, this maybe a single recommended object. More commonly, the recommendation engine constructs a list of objects with pointers or links to those objects from

which the user can choose. In principle, the recommendation engine could send back other forms or combinations of recommendations: people to contact, items to buy, places to go, etc.

The main issue is how to construct the recommendations. For example, the engine can use data mining techniques to try to estimate what the user would find interesting or useful or to find significant relationships in the opinion data (or between the opinion data and the objects in the object database). Current systems tend to be rather simple statistically (e.g., correlation among users) because most depend upon a very sparse set of user opinions to make recommendations. However, as the number of users increases, and data becomes less sparse, more sophisticated techniques will become possible.

In addition, even within well-populated systems, the data will not be uniformly distributed among subject areas or other divisions, so that a range of techniques will probably be needed. Some topic areas will always be sparsely populated because only a few people find them interesting or useful. Simple, non-parametric measures may be required for these areas. For areas where large amounts of opinion data are available, however, sophisticated algorithms will be more appropriate and desirable.

The recommendation engine may be connected to other sources of data or other systems. For example, it might get a list of pages from a search engine, and rank order the results by usefulness. It might share information with other recommendation engines. It might manage processes that get documents from a wide range of different sources.

2.4 Opinion Database

The *opinion database* contains the opinions collected by the opinion gatherer as well as metadata about the opinions. In more sophisticated systems, it may contain pre-calculated analyses of the opinion database to improve performance. The primary design issues here concern database performance. However, some secondary issues are also important. For example, should old opinions count as much as recent ones? Should they be discounted, and if so, at what rate? How does one handle opinions about dynamic objects (e.g., since a dynamic Web page may contain very different contents at different times, how should the opinion of the user about one set of contents generalize to the contents at other times)?

2.5 Object Collection

The *object collection* contains multimedia objects as well as metadata about these objects. In current systems, the objects are primarily text documents, such as movie or record or book titles, or descriptions of Web pages. However, any multimedia object could be rated and recommended. Future systems may, for example, recommend pictures, works of art, catalog items, sound bytes, stocks, or enterprise database objects, and deliver these directly to the user (e.g., not only recommending a movie but showing it also). Even more sophisticated systems may provide users with recommendations for components that are then integrated to meet the user's needs; for example, build-to-order software, or custom graphic designs based upon the user's preferences, or custom music compositions. In such systems, the interaction between user and recommendation system will undoubtedly be much more iterative. In any case, metadata about the object collection's components is an important design consideration, since it gives the recommendation engine the means to select specific objects, group them conveniently, or provide summaries about them to users.

This general model of a recommender system provides a framework for building a specific implementation. We now turn to our implementation, SurfAdvisor, which illustrates the set of design decisions and tradeoffs we made in constructing a recommender system.

3. SurfAdvisor

SurfAdvisor was designed to make recommendations about World Wide Web pages. As the user "surfs" the Web, SurfAdvisor provides recommendations for additional pages that the user might find interesting. In addition, it provides several types of information as the user moves the mouse over links on the Web page, including abstract, link speed, and ratings of the usefulness of the link. The overall structure of the system is shown in Figure 2.

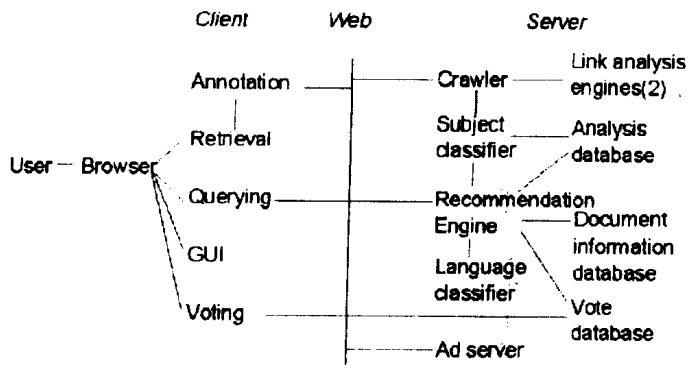

Figure 2

Both the client and server sides of SurfAdvisor use the Web Browser Intelligence (WBI) intelligent proxy system [2,3]. On the client side, the proxy intercepts all incoming Web pages and annotates them with JavaScript. This JavaScript causes the browser to display a separate window (the "minibar") which the SurfAdvisor user interface (see Figure 3). The minibar displays recommendations, abstracts, link speeds, and user votes. It also allows the user to vote. It does this without changing the appearance of the Web page the user requested. On the server side, another proxy distributes requests from the clients to the various components of the server and also generates timed requests to update the recommendations and run the crawler, as necessary.

Figure 3.

3.1 Opinion Gatherers

We implemented three sorts of *opinion gatherers* in SurfAdvisor. Two perform off-line *link analysis*, and the other is a *real-time voting system*. We discuss each in turn.

(1) Hubs and authorities. The first off-line method is that of Kleinberg [10] and Chakrabarti et al. [4], in which Web links were analyzed to find "hubs" and "authorities" for topics on the Web. Using this approach, a small set of Web pages on a given topic was used as a starting point. These were selected by querying a commercial search engine about the topic. All Web pages that were linked to on these starting pages were examined, and the pages that these linked to were examined. Using this set of pages around the starting pages, a weighting scheme for finding common groups of pages quickly converged on two key clusters: hubs, which may be thought of as reviews of the topic; and authorities, which are the web pages cited most often by the key reviewers. SurfAdvisor utilizes only the authorities, creating a list of these for each topic of interest.

(2) Citation frequency. The second off-line approach examined Web pages retrieved by a crawler from IBM's internal and external servers. The complete linkage map used to determine the most-frequently-cited sites on each topic. The lists of most-frequently-cited IBM pages became the second type of recommendation.

(3) User voting. The on-line method for gathering opinions allowed users to vote about each page. Users were asked to evaluate the usefulness of each page viewed on a scale of 1 to 4, with 4 representing most useful. The definition of useful was purposefully not given, so that users judged what was useful to them. User votes were stored in the opinion database. Two types of recommendations were calculated from these votes, as outlined in the next section.

Rationale. We chose to use more than one opinion gatherer because we wanted to provide recommendations from the first time the system was used, rather than waiting until the opinion database was sufficiently populated with votes. We chose link analysis as an initial form of pre-published opinions on Web documents that was sufficiently dense to be worth mining. We assumed, in effect, that the process of creating a Web link is sufficiently burdensome that almost all links point to pages that the creator of the links found useful. The "hubs and authorities" link analysis explicitly tests this assumption by looking for authorities that are cited by multiple hubs. In the case of links from within our own company, we felt that a simple link count would suffice, as there are relatively few "hubs" pages created by individuals to express their opinions of other internally-produced pages. Over the long term, however, we continue to believe that the explicit,

dynamic nature of voting on Web pages will produce recommendations that are superior to any static analysis of linkage or citations.

3.1.1 Opinions Collected from Named Individuals

SurfAdvisor collects opinions from users who are known by name to the system. Therefore, we had to design specific ways of ensuring frequent and reliable voting.

(1)Privacy. We ask new users to read and sign a privacy policy that describes what will be done with the information they provide. Our policy is to collect the user's name and e-mail address at registration. We then record their explicit votes in our database with an identifying number. We track sites visited and recommendations used, but with no information identifying users. We do not attempt to collect the links followed by a user, nor the amount of time spent at a particular site. We allow them to remove the information at any time. We have an explicit policy against releasing any personal information (including votes) to anyone. We have intentionally decided not to allow users to retrieve their previous votes, because this would leave open the possibility that others could masquerade as the user and retrieve this personal information.

(2)Ease of rating. We have chosen a mechanism that allows users to vote with a single mouse click on a bitmap image that is always visible as they surf the Web (Figure 3). Users vote on the page that is currently in the browser. Framesets presented somewhat more of a difficulty: the decision is whether to record the vote for the current frameset or the current frame in a page. Each of these has its drawbacks. A frameset may be in effect a dynamic document which has a constant URL but a varying set of frames. On the other hand, the "current" frame is often simply a matter of the order of loading of the individual frames and thus may be almost anything. We have chosen to record the frameset for purposes of voting, but to determine the subject of the frameset each time a new user visits it. Thus, SurfAdvisor sees the same text as does the user, and the vote gets recorded with the newly-determined subject.

(3) Rewarding the user. We reward the user with increasingly personalized recommendations about pages on the Web.

Rationale. Some recommender systems utilize measures such as number of visits by a user to a particular Web page or the amount of time a user spends viewing a Web page. This is also technically possible with SurfAdvisor but we have chosen to rely only on votes. The major reason for this is our concern for user privacy. We think many users will shy away from systems that keep records of every site visited, even with assurances about anonymity. Hence, the only information we keep associated with an individual is the record of votes. In this way, the user controls on a page-by-page basis exactly what information will be kept. For the same reason, we gather no demographic or other personal information. We also believe that explicit votes are a more reliable measure of the usefulness of a particular page. The tradeoff, of course, is that users vote on only a small fraction of the pages they visit, meaning that we have less data to mine.

3.2 Query Generator

SurfAdvisor has an automatic *query generator*. When the user's browser loads a new page, this triggers (via JavaScript) a query to the recommendation engine, which in turn produces a short list of recommended pages on the same topic as the page currently loaded by the browser. A complete list, including abstracts, can be obtained by a single click on the minibar. In addition, when the user mouses over a link, several queries are generated. These retrieve and display the following: (a) an abstract of the page pointed to by the link; (b) average user rating of the linked-to page; (c) previous ratings by the user; and (d) the speed to the link's server. The user's votes and link speed are computed locally to improve performance. All of these are displayed asynchronously (typically in less than 1 second).

The query generator automates the process as much as possible. Although this may result in less tailored queries than in manual systems, we found in our early pilots that users liked our method.

3.3 Recommendation Engine

The SurfAdvisor *recommendation engine* delivers four types of recommendations, all tied to the topic of the page the user is currently viewing. The first type of recommendation is the pre-computed set of links that are most highly linked to by other IBM pages. This is computed simply by identifying the subject of each page (see below) and keeping a database of linkage information about all of the pages visited by the crawler. Those pages on a given topic with the most links from other IBM sites are tabulated and stored. The second type of recommendation is the pre-computed "authorities", as described previously. The third type is a list of those pages on a given topic receiving the highest number of votes by all users. This is computed periodically to reflect recent voting information. The fourth type is the set of pages "recommended by a virtual friend." In this case, all votes by all users within the desired topic area are examined to determine the voters whose voting patterns are most similar to the user's. These voters become the user's "virtual friends." Pages deemed highly useful by the virtual friends, but not yet voted upon by the user, are delivered as the recommendations.

When the user requests a set of recommendations, up to 50 recommendations of each of the four types are delivered. To improve the performance in offering these recommendations, we cache all of them (except the virtual friends set, which are computed in real-time) in memory.

3.4 Opinion Database

The *opinion database* contains all votes (each of a scale of 1 to 4), and is segmented by topic area. Within a given area, any votes by users are recorded, along with a time stamp. The time stamp allows the votes to be discounted as time passes. Segmenting opinions by topic allows us to improve access speeds to the data, particularly for finding the "virtual friends" of a given user.

3.5 Document Information Database

Information about each document known to the system is stored in the *document information database*. Documents become known to SurfAdvisor in two ways: either the background crawler, which is continuously running, finds the page, or the page is visited by a user. For dynamic Web pages, the information in the database is updated upon each visit by a user. This information includes the URL, subject classifications (see below), the average vote to date, title, abstract, and so on. There is one entry for each page or link ever visited; a separate table is used to maintain information about the URLs that have "moved" or changed location.

We are careful not to store the abstracts of dynamic pages for privacy reasons. Some queries to such sites contain information about the user, and hence the page resulting from the query may also.

3.6 Document Classification

Each document, whether it was obtained by the crawler or by a user, is classified by subject and optionally by language. The *language classifier* is that of Prager [13]. To do the subject classification, we have created a *taxonomy* using a supervised clustering technique described elsewhere [1]. The clustering used a training set of categories and a set of example documents of each; these training data were derived from several sources, including manual selection. After clustering, we examined the resulting categories to ensure that they were sufficiently distinct and to assign a label to each of them. This process resulted in 1167 clusters or subject areas, each described using a vector to the pseudo-centroid of the cluster. These vectors are cached in memory for rapid computation.

A *subject classifier* that uses the same model as that of the taxonomy rapidly classifies the subject of the page by computing the distance from the document to each pseudo-centroid. The nearest three subjects of each page are recorded in the document information database, along with the distance measure for each. When recommendations are presented to the user, the three subjects are shown, so that the user may select a different but related subject.

3.7 User Interface

The user interface, as shown in Figure 3, is a separate browser window. In the upper left corner is the area in which users can vote. In the upper middle are the ratings and speed indicators that are activated when a user mouses over a link. On the right, the ad or information item is displayed (a place-holder for an ad is pictured); this space is also used for the abstracts of links. On the bottom left is the button that produces a complete listing of recommendations for the user. The user is permitted to change the topic of the recommendations to any of the three top choices of subject determined by the subject classifier. In the lower middle is the short form of the recommendations. On the bottom right are buttons to allow the user to get on-line help or see our privacy policy. JavaScript that has been added to the document by the intelligent proxy permits the minibar to detect "mouseover" events in the main browser page. It also displays the ads, abstracts, ratings, and speeds as appropriate by issuing queries to the server or the client databases.

Rationale. In our earliest pilots studies, we tried placing the information now in the minibar on the current browser page, rather than in a separate page. Users quickly told us they preferred to have the browser display the page with no additional annotations. Currently, in addition to the main browser window and the minibar, the recommendations are displayed in a third browser page.

3.8 Other Features

SurfAdvisor also delivers informational or commercial advertising to users, targeted to the subject matter of the page the user is viewing. This is seen by the user whenever the mouse is not over a link, or when we do not have an abstract for the link over which the user's mouse currently sits. An ad server is therefore also included. This ad server provides ads or information targeted at the current interests of the user.

4. Results and Discussion

We ran several small-scale pilots and one larger-scale pilot to test various features of SurfAdvisor on users. The larger scale pilot included approximately 130 users over a period of three months. We have assessed several different aspects of SurfAdvisor during these tests, including its ability to classify documents, its performance, user responses, and privacy.

(1) Ability to classify documents. During our earliest pilots, we found that the classification of documents was a primary user concern. Users expected each document to be assigned to the correct category, and they expected the category name to describe the Web page they were visiting with a fair amount of precision. Hence, we spent considerable time investigating taxonomies and classifiers. Our experience was that typical classifiers had two major shortcomings: they were generally too slow for real-time use, or they did not provide the accuracy or precision that users expected. Part of the problem is that most classifiers use a manually determined taxonomy built on a perception-based conceptual model as opposed to a mathematical model that could be directly used by a classifier. This problem is particularly acute for taxonomies designed for large, heterogeneous data collections. For example, Chakrabarti et al. report an error rate of 68% with a Yahoo! sample, which improved to 21% with the use of information about hyperlinked documents [5]. We therefore decided to build our own taxonomy, and then to build a classifier that utilized exactly the same model as the clustering process used to build the taxonomy [1]. We found that the resulting taxonomy was as accurate in describing Web pages as an equivalent Yahoo! taxonomy, and that the classifier, because it uses the same model as the clustering system used in building the taxonomy, has an accuracy limited only by the degree to which a given Web page falls outside of the class definition. Further, we can resolve even very closely related categories and even pages with multiple or compound topics (e.g., lawyer jokes) are correctly analyzed. A 400 MHz Pentium-based machine can retrieve and classify up to 10,000 URLs per hour (depending upon Web traffic). The rate is limited by the number of URLs that can be retrieved from the Web, rather than the classification process. Hence, it is possible to do real-time classification with this system.

(2) Performance. In addition to the performance of the classifier, the other performance factor is the amount of time needed to display recommendations. In our current system, this

consisted of two phases: annotating the Web page with JavaScript, which took less than one second for a typical page, and retrieving the recommendations, which can take up to 5 seconds or more if the page has not been previously classified. We have therefore designed the system to load the recommendations asynchronously, and to enforce a maximum wait of 5 seconds so that dead or slow URLs do not cause the user undue delay.

(3) User responses. During our three-month pilot, 134 users visited approximately 32,000 sites, or approximately 240 sites per user. Seventy-five of these users voted at least once. They voted on 936 or approximately 3% of these sites. At each site they received recommendations for about 50 URLs. Of all the sites visited, 2279, or 7% were ones recommended by SurfAdvisor. Users were given the opportunity to select from two other subject areas each time they were given a set of recommendations, but did so only 12 times (0.04%).

(4) Privacy policy. Users were asked to electronically sign a privacy policy. We did not monitor how many users opted out at that point, but had no cases of users complaining about the privacy policy or even suggesting improvements.

(5) Recommendations. The recommendations provided by SurfAdvisor proved to be very interesting. An example of this is shown in Table 1.

	User's Choices	**Authorities**	**Top IBM Sites**
First choice	VisualAge Developer Domain (ibm.com)	Java Technology Home Page (sun.com)	IBM Java Information Hub
Second choice	Java resources (ibm.com)	The Java Developers Kit (sun.com)	IBM Internal Java Centre
Third choice	comp.lang.java FAQ (unc.edu)	comp.lang.java FAQ (unc.edu)	Java Site Index

Table 1.

This set of recommendations was typical, in that users displayed different interests than those found by the automated methods we used. Here, for example, the users' first choice was a page on VisualAge that was recommended as one of the IBM sites, but as only the sixth most linked-to site. The second user choice was found entirely by the users. And the third choice was also the

third choice in our "authorities" list. Obviously, a non-IBM set of users might select a different set of top choices, but one again reflecting the specific needs of their own work or home interests.

5. Conclusions

Some key findings from our pilot tests which we believe generalize to other systems include:

(1) Users expect a considerable degree of specificity and accuracy in subject classification. We found that our taxonomy (of 1167 subjects for Web pages) is adequate to provide the necessary specificity, but we estimate that 5-10 times more subjects would meet with even more user approval.

(2) Users expect real-time performance. A system that does not deliver recommendations within a few seconds is perceived as being much less useful than one that does. Our current system delivers recommendations in less than a second if the page has been previously visited by a crawler, and in less than one second after the classification engine receives a new URL to classify. The user does not see any appreciable delay in the latter case because the new URL is requested by the recommendation engine at the same time the user's browser requests the URL. We found users to be very intolerant of anything that slowed the apparent performance of the browser.

(3) Pre-populating the recommendation database is important. Users want so see recommendations immediately, not after the system has been in use for some time. Hence, any such system needs either to have a pre-loaded set of recommendations (e.g., our "authorities"-based recommendations) or to have recommendations from raters who have been hired specifically to rate documents, or to use some similar scheme.

(4) Systems that depend upon user voting (such as SurfAdvisor) will typically see the voting patterns of individual users vary widely. Half of our participants never voted, and most of the votes were added by a small number of participants. The overall voting rate was about 5% of all URLs visited.

(5) Because of the relatively low voting rate, any system that depends upon explicit voting will need to carefully design the means by which voting patterns of users are compared. Even within popular subject areas, the voting patterns are sufficiently sparse that users rarely rate

the same sites. Single-topic systems, such as Firefly [20] have solved this problem by presenting users with a fixed set of documents to rate, and this is probably the correct model for multi-subject systems as well.

6. Future Directions

Recommender systems will undoubtedly continue to proliferate, primarily to help personalize shopping on the Web. However, this is a limited view of their usefulness. In the long term, the application of recommender systems may in fact be much broader in knowledge management, where they can help workgroups or others actually implement knowledge sharing. In this context, a recommender system can be thought of as a way of automating the sharing of knowledge about what is useful and where it can be found.

From a technology viewpoint, many of the challenges of future recommender systems lie in applying appropriate, efficient data mining techniques to improve the quality of recommendations. As the opinions databases of recommender systems get larger and larger, they become more and more important part of the intellectual capital of those who own the system, and hence become more worth mining. The primary commercial value of a recommender system, in fact, is not the system itself, but the database of information about users that can be used to competitive advantage to tailor advertising, products, and services. Future research in the area of efficient algorithms for finding relationships among users or objects will considerably improve the value of these systems to both users and owners.

Some of the important needed technology changes are in the area of user interface design for these systems: current systems often have very primitive user interface designs. Better UIs will help encourage even naive users to interact with recommender systems. And since recommender systems offer personalized recommendations, can personalized UIs be far behind?

Another significant area for future research in recommender systems is in how to control what information gets shared. For example, information shared with members of a workgroup may be totally inappropriate for sharing with a general audience. Although there are those who assert that there is no privacy any more [12], implementing privacy policies about the data may be a significant design consideration both in Europe and elsewhere.

One interesting, but unstudied aspects of recommender systems is the effect that they have on their users. We suspect, for example, that widespread use of recommender systems on the Web may have the interesting effect of improving the quality of Web pages. A Web page author who sees that everyone who visited her or his site rated the page as non-useful may behave very differently than an author who merely sees that 500 people visited the site this month. The quality of Web pages, in our opinion, is related very directly to the lack of real feedback on the Web. To the extent that recommender systems provide public ratings of Web pages, they can give feedback to authors. More sophisticated authors could even use such ratings to track user reaction to their Web pages as the content or design of the pages changes.

7. Summary

Our experience so far with recommender systems suggests that users perceive a strong need for recommendation systems in complex environments such as the Web. However, users desire recommendations that are as good and as personal as they would get from a friend. The challenge in building recommendation systems is to get that quality of recommendation and to deliver it in a way which seems easy and natural to the user. SurfAdvisor does this.

8. References

[1] Aggarwal C. C., Gates S C., Yu, P.S. On the merits of building categorization systems by supervised clustering. Submitted to SIGKDD-99.

[2] Barrett, R. and Maglio, P. P. Intermediaries: New places for producing and manipulating web content. *Computer Networks and ISDN Systems*, 30 (1998), 509-518.

[3] Barrett, R., Maglio, P. P., and Kellem, D. How to personalize the web. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '97).* New York, NY: ACM Press (1997).

[4] Chakrabarti, S., Dom, B. E., Gibson, D., Kumar, R., Raghavan, P., Rajagopalan, S. and Tomkins, A. Spectral Filtering for Resource Discovery. *Proceedings of the SIGIR 98 Workshop on Hypertext Information Retrieval for the Web.* Editors: Eric Brown and Alan Smeaton. Also

http://lorca.compapp.dcu.ie/SIGIR98-wshop/Tomkins.ps.

[5] Chakrabarti, S., Dom, B., and Indyk, P. Enhanced hypertext categorization using hyperlinks. *Proceedings of the ACM*, SIGMOD, 1998.

[6] Dyson, E. *Release 2.0 : A Design for Living in the Digital Age.* Broadway Books; ISBN: 0767900111, 224pp, 1997. See, for example, her discussion of anonymous participation in electronic discussion groups.

[7] Goldberg, D., Nichols, D., Oki, B.B., and Terry, D. Using collaborating filtering to weave an information tapestry. *Communications of the ACM* 35, 12 (Dec. 1992), 61-70.

[8] Kautz, H., Selman, B., and Shah, M. Combining social networks and collaborative filtering. *Communications of the ACM* 40, 3 (March 1997) 63-65.

[9] Keller, R.M., Wolfe, S.R., Chen, J.R., Rabinowitz, J.L., and Mathe, N. A Bookmarking Service for Organizing and Sharing URLs. Proceedings of the 6th Intl. WWW Conference, Santa Clara, CA, April, 1997.

[10] Kleinberg, J. Authoritative sources in a hyperlinked environment. *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 1998 and http://simon.cs.cornell.edu/home/kleinber/auth.ps

[11] Konstan, J.A., Miller, B.N., Malt, D, Herlocker, J.L., Gordon, L.R., and Riedl, J. GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM* 40, 3(March 1997) 77-87.

[12] McNealy, S. Quoted in *The New York Times*, p. A1, March 3, 1999 as saying "You already have zero privacy -- get over it" in an article by John Markoff.

[13] Prager, J.M., Linguini: Language Identification for Multilingual Documents. *Proceedings HICSS-32*, Maui, Hawaii, 1999.

[14] Resnick P. , Varian H.R. Recommender systems. *Communications of the ACM* 40, 3(March 1997) 56-58.

[15] Terveen, L., Hill, W., Amento, B., McDonald, D., and Creter, J. PHOAKS: A system for sharing recommendations. *Communications of the ACM* 40, 3(March 1997) 59-62.

[16] http://point.lycos.com/categories/

[17] http://www.alexa.com

[18] http://www.alexa.com/whatisalexa/privacy_policy.html

[19] http://www.amazon.com and link at bottom of page on privacy policy.

[20] http://www.firefly.com

[21] http://www.lycos.com/privacy/

[22] http://www.yahoo.com

[23] http://www.yahoo.com/info/suggest/