Collaborative value filtering on the Web

Authors omitted

Submission Identification Number: 180

1. Rationale

Today’s Internet search engines help users locate information based on the textual similarity of a query and potential documents. Given the large number of documents available, the user often finds too many documents, and even if the textual similarity is high, in many cases the matching documents are not relevant or of interest. Our goal is to explore other ways to decide if documents are "of value" to the user, i.e., to perform what we call "value filtering."

In particular, we would like to capture access information that may tell us-within limits of privacy concerns-which user groups are accessing what data, and how frequently. This information can then guide users, for example, helping identify information that is popular, or that may have helped others before. This is a type of collaborative filtering or community-based navigation [2].

Access information can either be gathered by the servers that provide the information, or by the clients themselves. Tracing accesses at servers is simple, but often information providers are not willing to share this information. We therefore are exploring client-side gathering. Companies like Alexa [7] are currently using client gathering in the large. We are studying client gathering at a much smaller scale, where a small community of users with shared interest collectively track their information accesses. For this, we have developed a proxy system called the Knowledge Sharing System (KSS) that monitors the behavior of a community of users. Through this system we hope to:

1. Develop mechanisms for sharing browsing expertise among a community of users; and

2. Better understand the access patterns of a group of people with common interests, and develop good schemes for sharing this information.

2. Implementation details

The KSS system is composed of two prototypes: the KSS proxy and the KSS metasearch engine.

The KSS proxy is a web proxy that not only caches web pages but also tracks the URLs accessed by users. It then uses this information to "rank" the links embedded in the web pages that are returned to the user. The KSS proxy intercepts URL requests from a client, stores the URL in a KSS-URL database for future reference, and forwards the HTTP requests to the target web server. The resulting web page is also intercepted, cached and parsed. The parsing process selects those URLs in the web page that are already in the KSS-URL database. It modifies the corresponding link texts to represent the fact that the respective links have already been accessed. This is done by adding to the link text the number of times anybody in the group has accessed that link. Furthermore, the system generates a full text index of all the pages in its cache. Users can later query the index, using a local search engine.

The second component of the KSS system, the KSS metasearch engine, performs a similar task but in the context of interactions with search engines such as Altavista, Lycos or Excite. Operation is similar to that of any metasearch engine. However, the KSS engine performs two additional tasks: it attempts to find the best search engine(s) for a given query, and it rank-merges result hit lists returned when a query is submitted to multiple engines.

Because the KSS metasearch engine mediates the interaction between the user and the search engines, it can remember the queries asked, and it can remember which result hits users have followed for each query in the past. This in turn allows the metasearch engine to recommend the best search engine to use when queries are resubmitted later on, presumably by other users.

Analogous to the KSS proxy, the KSS metasearch engine annotates the links in the result hit lists: The number of times each result URL has been used in the past is added to the link text. In addition, the number of previous accesses is used to rank result lists merged from multiple search engines. Both systems, the KSS proxy and the metasearch engine, can work together, so the data stored in the KSS URL database can be used to sort the set of links in the metasearch engine.

The KSS proxy uses a modified Harvest cache [3] and the metasearch engine is a collection of cgi scripts in msqltcl, an extension of the Tcl interpreted language.

3. Distributed KSS

The KSS system has been designed to serve a small community of users taking advantage of their common informational interests. However, if we limit the scope of the system to these small groups we could be missing interesting data from other user communities on the web. On the other hand, if we simply interconnect several KSS systems and merge all their databases, we could have a completely unrelated set of links that could mislead users. As a compromise, we have designed a federation of KSS systems, where each system is centered on a small community, but, periodically, some of these KSS systems acquire each other’s databases. The decision of acquire another server’s database is made based on a comparison of database content summaries provided by each server. Periodically, each server examines the summaries of the other servers to determine the database that is most closely related in topic to its own. The remote data is stored separately from the local access database, so it will not bias that local information.

The goal of this federation mechanism is to build an environment of collaborative KSS systems, but without limiting the independence of the participants. It uses the concepts developed for other federated environment such as Aleph [4]

4. Collecting usage statistics

The KSS system has been tested for two months, by members of the Stanford Digital Library Project and by members of the Philips Research Lab in Palo Alto. We are starting to collect usage statistics, which we hope will eventually guide the design of systems such as KSS. For example, one statistic we have already analyzed is the average fraction of links in pages (returned to users) that are in the KSS URL database. After two months of usage, this average was found to be 22.49%. This means that in an average page presented to the user, over a fifth of the links will be annotated with the number of previous community accesses. The rest of the links will not be annotated. This number is very close to the hit rate hit of a normal web caching proxy [5,6], which is around 30%. But it is not clear why the two numbers are similar. The 22.49% value is high enough that KSS users are getting usage information for a significant number of links. We expect to obtain other statistics such as this one in the near future.

5. References

[1] G. Abdulla, M. Abrams, E. A. Fox, *Scaling the WWW* Technical Report TR-96-06, Computer Sci. Dept. Virginia Tech, March 1996

[2] Kent Wittenburg, Duco Das, Will Hill, Larry Stead *Group Asynchronous Browsing on the World Wide Web* . Issue 1 of the WWW Jounal.

[3] C.M. Bowman, P.B. Danzing, D.R. Hardy, U. Manber, and M.F. Schwartz. *The harvest information discovery and access system* In Proc. of the Second International World Wide Web Conference, Chicago, Illinois, October 1994.

[4] L. Navarro, G. Rodriguez *Aleph: a new computational model on the web.* In P. Proc. of the 5th Internationl World Wide Web Conference, Paris, France, May 1996

[5] Marc Abrams, ed altres. *Caching Proxies: Limitations and Potentials* Proc. 4th Inter. World-Wide Web Conference, Boston, MA, Dec 1995

[6] G. Abdulla, M. Abrams, E. A. Fox, Stephen Williams *WWW Proxy Traffic Characterization with application to caching* Technical Report TR-97-03, Computer Sci. Dept., Virginia Tech, March 1997

[7] Alexa. http://www.alexa.com